AD-A270 426

DTIC
ELECTE
OCT 04 1993
S A D

Final Report

Volume 1

Development of Parallel Architectures For Sensor Array Processing

Submitted to:

Grant No. N00014-91-J-1011
Department of the Navy
Office of the Chief of the Naval Research
Arlington, VA 22217-5000

This document has been approved for public release and sale; its distribution is unlimited.

Submitted by:

M. M. Jamali, Principal Investigator
S. C. Kwatra, Co-Investigator

Department of Electrical Engineering
College of Engineering
The University of Toledo
Toledo, Ohio 43606

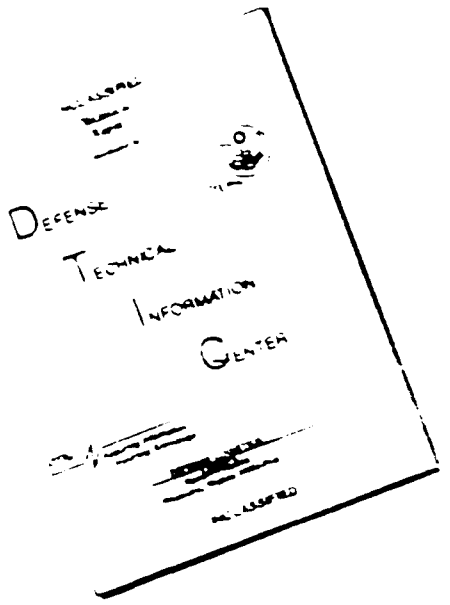
Report No. DSPH-3
August 1993

93-23066

93

Best
Available
Copy

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

Final Report

Volume 1

Development of Parallel Architectures For Sensor Array Processing

DTIC QUALITY INSPECTED 2

Submitted to:

Grant No. N00014-91-J-1011
Department of the Navy
Office of the Chief of the Naval Research
Arlington, VA 22217-5000

Accesion For
NTIS CRA&I
DTIC T-B
Unannounced
Justification
By A253680
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

Submitted by:

M. M. Jamali, Principal Investigator
S. C. Kwatra, Co-Investigator

Department of Electrical Engineering
College of Engineering
The University of Toledo
Toledo, Ohio 43606

Report No. DSPH-3
August 1993

This report contains the work performed under ONR grant N00014-91-J-1011 during the period October 1990 to July 1993. Two earlier semi annual report submitted in summers of 1991 and 1992 are also part of this work. Dr. A. H. Djouadi research associate, Rao, Sheelvant, Surya and Tabar graduate assistants contributed to this work.

M. M. Jamali

Principal Investigator

DEVELOPMENT OF PARALLEL ARCHITECTURES FOR SENSOR ARRAY PROCESSING

The high resolution direction-of-arrival (DOA) estimation is important in many sensor systems. It is based on the processing of the received signal and extracting the desired parameters of the DOA of plane waves. Many approaches have been used for the purpose of implementing the function required for the DOA estimation [1-7]. The Multiple Signal Classification (MUSIC) [1] and the Estimation of Signal Parameters by Rotational Invariance techniques (ESPRIT) [2] algorithms are two novel approaches used recently to provide asymptotically unbiased and efficient estimates of the DOA. They are believed to be promising and appropriate for hardware implementation for real time applications. They estimate the so called signal subspace from the array measurements. The parameters of interest (i.e. determining of the DOA) are then estimated from the intersection between the array manifold and the estimated subspace.

Although MUSIC is a high resolution algorithm, it has several drawbacks including the fact that complete knowledge of the array manifold is required, and that is computationally very expensive as it requires a lot of computations to find the intersection between the array manifold and the signal subspace. The drawback of high computational requirements can be overcome with the use of highly parallel systems which will perform computation in real time.

Design of special purpose hardware for the implementation of various real time algorithms is possible due to advances in VLSI technology. The customized hardware has two main advantages as listed below:

1) The given algorithm is executed at a high speed.
2) Cost and size of the hardware will be lower than the cost of a general purpose computer.

Pipeline, parallel and distributed processing approaches can be exploited to achieve high throughput rates. In order to develop a real time parallel architecture for a given algorithm, following steps need to be performed.

1. An algorithm should be first converted into a computationally efficient algorithm.
2. The selected algorithm is divided into parallel modules.
3. If a particular module of the algorithm can not be executed in parallel due to data dependency it should be pipelined.
4. After parallelizing the algorithm, it should be mapped on a suitable architecture.

There are many architectures such as systolic arrays, Single Instructions Multiple Data (SIMD) systems and Multiple Instructions Multiple Data (MIMD) systems which can be used for parallel implementation [8-11]. There are cordic processors available in the literature which can also be exploited. An appropriate structure which can exploit maximum parallelization to reduce the computation time will be selected for real time implementation for the particular application.

This approach of designing special purpose hardware has been applied to the high resolution direction -of -arrival (DOA) estimation for narrowband and wideband cases. The DOA estimation algorithms are based on the processing of the received signal and extracting the desired parameters of the DOA of plane waves. Many approaches have been used for the purpose of implementing the function required for the DOA estimation and are available in the literature [1-7]. After combing the literature thoroughly, MUSIC algorithm was selected to develop special purpose hardware for real time computation. Summary of the MUSIC algorithm is as follows:

1) Estimate the data covariance matrix.

2) Perform the eigendecomposition.

3) Estimate the number of sources.

4) Evaluate Power function.

5) Find the d largest peaks of power to obtain estimates of the parameters.

First of all MUSIC algorithm has been modified with efficient computational modules. The algorithm has been parallelized. Four modules are implemented using pipeline and parallel processing schemes. First module will compute the covariance matrix. The second and third modules will compute eigenvalues and eigenvectors which will be used by the fourth module. This module computes the power function giving the desired DOA.

The sampled data obtained from the sensors is used to obtain the data covariance matrix. Since the covariance matrix is Hermetian, the computation of lower triangular matrix of covariance matrix is sufficient to get complete information of the full matrix. It is well known that the symmetric eigendecomposition problem is one of the fundamental problems in signal processing as it arises in many applications such as DOAs estimation and spectral estimation. Most methods reduce the problem to the generalized eigendecomposition problem by computing the data covariance matrix. Householders method is a technique used for reducing the bandwidth of the data covariance matrix by transforming it to a tridiagonal one under congruent transformations without affecting the values of the eigenvalues [12].

In order to transform the m x m data covariance matrix to a tridiagonal matrix, m-2 Householder transformations are performed. Each transformation is determined to eliminate a whole row and column above and below the subdiagonals without disturbing any previously zeroed rows and columns. QR method is used to convert the tridiagonal matrix to a diagonal matrix which gives the eigenvalues and eigenvectors. The Eigenvalues are used to find the

number of sources and finally using the eigenvectors in Power method we find the angle of arrival. Details for developing hardware for the MUSIC Algorithm are given in [13-14]. The Hardware block diagram for ESPRIT algorithm is given in [13-14]. It is similar to the MUSIC algorithm but instead of power method of MUSIC some more computations are performed to evaluate the angle of arrival in case of ESPRIT.

For the wideband case, there are many algorithms which are available in the literature. Some of them are extensions of the narrowband cases and others are developed for wideband cases. After reviewing the current literature two wideband approaches namely Broad-Band Signal-Subspace Spatial-Spectrum (BASS-ALE) Estimation algorithm [6] proposed by Buckley & Griffiths and bilinear transformation algorithm [7] proposed by Shaw & Kumaresan have been selected for hardware implementation. These algorithms were simplified, pipelined and parallelized. The BASS-ALE algorithm, its modifications, hardware implementation and generalized algorithms both for narrowband and wideband have been described in Volume 2 of this report. The bilinear transformation algorithm, its modifications, hardware implementation and VLSI implementation of generalized covariance processor have been described in Volume 3 of this report.

The goal of this work was to design an architecture which will be suitable both for narrowband and wideband cases. It has been discovered that the narrowband MUSIC algorithm and the wideband algorithms BASS-ALE and bilinear transformation requires similar computational modules such as the computation of the covariance matrix, eigenvalues and eigenvector computation using the Householder transformation and QR method and power method [13-14]. Since the required modules are identical, they have been generalized into one algorithm and generalized hardware is developed. The generalized hardware will be suitable for both applications.

In these DOA applications first the data has to be collected by the sensors to compute the covariance matrix. In this study eight sensors and eight delay elements have been assumed and hardware is designed accordingly. Eight sensors in the narrowband case will result in a 8*8 covariance matrix. Therefore all computations for DOA estimation will require manipulation of 8*8 matrices. If eight PEs are used in the architecture it would be easier to map the algorithm on these eight PEs. Therefore four modules each using eight PEs can be utilized for this purpose. They will operate in parallel and pipeline fashion.

For the case of the wideband, the BASS-ALE algorithm also requires eight delays in the sensor array. Using eight sensors and eight delays will result in a data vector of size 64. Therefor the computation of the covariance matrix will involve 64*64 matrix. Moreover the Householder transformation, QR method and power method will all operate on 64*64 matrices. It is desired to use eight PEs in each module as that will be sufficient and will also satisfy the real time requirement. Use of 64 PEs will simply result in an efficient use of the PEs. Use of eight PEs in the manipulation of 64*64 matrices will require proper mapping of the algorithms on these arrays. It is proposed that the covariance matrix computation, Householder transformation , QR method and power method all use modules with eight PEs. Architectures of various modules are described in detail in previous semi annual reports, and Volumes 2 and 3 of this report.

WORK PERFORMED

1. A literature survey has been performed and investigated for various algorithms available for narrow band and wide band cases.

2. In the area of narrow band, MUSIC and ESPRIT algorithms were selected and further studied. These algorithm were converted into computationally efficient algorithms and subsequently parallelized. Three different architectures namely Systolic architecture, Cordic processors and SIMD were considered as reported in [13].

3. These algorithms required eigenvalues decomposition. Householder transformation was used to convert covariance matrix into tridiagonal matrix. The QR method was used to finally obtain eigenvalues and eigenvectors. The detailed parallel architecture has been developed for parallelized Householder transformations and QR method and has been reported in [13-14].

4. An architecture for a generalized processing element suitable for sensor array processing elements has been developed. Its custom instruction set has been developed and is given in Volume 2.

5. Single instruction multiple data (SIMD) type of architecture lend themselves for the implementation of narrow band DOA estimation. The computation of covariance matrix, Householders transformation and QR method can be easily computed using SIMD machine. The work on SIMD machine has been performed and is given in Volume 2.

6. In the case of wideband DOA estimations, various algorithms available in the literature were studied. It was found that wideband DOA estimation is more computationally intensive than narrow band case. An algorithm proposed by Shaw has been selected for further study. This

algorithm again has been modified and substituted with computationally efficient operations. An architecture for the computation of this algorithm is developed and is described in Volume 3.

7. DOA estimation for wide band sources using "Broad-Band Signal- Subspace Spatial Spectral (BASS-ALE) estimation algorithm has been studied, simplified, parallelized and its architecture has also been developed and is reported in Volume 2.

8. To verify the validity of this work, following simulation programs are written.

(a) Data generation of narrowband and wideband sources for eight sensor arrays has been computed.

(b) Simulation of MUSIC algorithm and DOA estimation is completed.

(c) Simulation of BASS-ALE algorithm for wideband sources is almost complete.

(d) Simulation of DOA estimation using bilinear transformation approach.

(e) Simulation program to study quantization effects.

9. A simulation has been performed at architecture level using VHDL software to check computational complexity of the combined covariance matrix processor. Logical level and circuit level design was done using Viewlogic's computer aided design (CAD) tool Powerview. This circuit level design was used to layout a Very Large Scale Integration (VLSI) chip for combined covariance matrix processor using Mentor Graphics GDT tools.

10. As the entire architecture cannot be accommodated on a single chip, multiple modules are used. This involved study of inter-chip communication, I/O bus architecture for each chip, bus arbitrator etc.

11. A single Generalized Processor (GP) has been developed which minimizes cost and design time. It uses micro coded architecture and has specialized logarithm unit. The details of this generalized processor are given in Volume 2.

12. A study has been performed to estimate real time requirements for the computations of DOA. Based on this study, real time architectures for the computation of narrowband MUSIC and two wideband algorithm architectures have been developed. They all use parallel modules and each module will have eight processing elements. This arrangement will meet real time computation requirement for estimating upto seven sources in sonar environment.

13. Parallel architecture for BASS-ALE algorithm for wide band sources has been developed and is given in Volume 2.

14. Parallel architecture for bilinear transformation algorithm for wide band sources, has also been designed and is given in Volume 3.

15. Copies of various papers that resulted from this work and presented at various conferences are given in the appendix.

REFERENCES

1. R.O. Schmith, "Multiple emitter location and signal parameter estimation," IEEE Trans. on Antennas and Propagation, Vol AP-34, No.3, PP. 276-280, Mar. 1986.

2. R. Roy and T. Kailath, "ESPRIT-Estimation of signal parameters via rotational invariance techniques," IEEE Trans. Acoustic, Speech and Signal Processing, Vol. 37, No. 7, PP 984-995, July 1989.

3. D. Spielman, A. Paulraj, "Performance analysis of the MUSIC algorithm," in ICASSP 1986., PP. 1909-1912.

4.T. J. Shan and A. Paulraj, "On smoothed rank profile tests in eigenstructure approach to direction-of-arrival estimation," ICASSP 1986, PP 1905-1908.

5.C.Y.Chen and J. A. Abraham, "Fault-tolerant systems for computations of eigenvalues and singular value" SPIE Vol. 696, Advance Algorithm and architecture for signal processing, pp 228-237, 1986.

6. Kevin M. Buckley and Lloyd J. Griffiths, "*Broad-band signal- subspace spatial-spectrum (BASE-ALE) estimation*", IEEE Trans. on Acoustics, Speech, and Signal Processing, VOL. 36, No. 7, July 1988.

7. Arnab K. Shaw and Ramdas Kumaresan, "*Estimation of angles of arrivals of broadband signals*", IEEE ICASSP-87, pp.2296-2299, 1987.

8. S.Y. Kung, "VLSI Signal Processors," Prentice Hall 1987.

9. C.F.T. Tang, K.J.R. Liu, S.A. Tretter, "On systolic array for recursive complex Householder transformations with applications to array processing," ICASSP 1991, PP 1033-1036.

10. K.J.R. Liu, S.F. Heieh, K. Yao, "Two level pipelined implementation of systolic block Householder transformation," ICASSP 1990, PP. 1631-1634.

11. W. Phillips and W. Robertson, "Systolic architecture for symmetric tridiagonal eigenvalue problem," IEEE International Conference on systolic arrays, PP. 145-150, 1988.

12. K.J.R. Liu, K. Yao, "Multiphase systolic architecture for spectral decomposition," 1990 International conference on parallel processing, PP I-123-I-126.

13. M. M. Jamali, S.C. Kwatra, "Development of parallel architectures for sensor array processing algorithms. "Report No. DSPH-1, University of Toledo, 1991.

14. M. M. Jamali, S.C. Kwatra, "Development of parallel architectures for sensor array processing algorithms. "Report No. DSPH-2, University of Toledo, 1992.

APPENDIX

Copies of the papers presented at various conferences

ICSPAT

The International Conference on Signal Processing Applications and Technology

BOSTON '92

November 2-5, 1992

A Parallel Architecture for MUSIC Algorithm

R. B. Sheelvant, M.M. Jamali, A. H. Djouadi, S.C. Kwatra
Department of Electrical Engineering
The University of Toledo
Toledo, OH 43606
419-537-2580

ABSTRACT

The high resolution direction- of-arrival(DOA) estimation is important in many sensor systems like radar, sonar and seismic exploration. Multiple signal Classification algorithm for DOA estimation has been studied to develop a parallel architecture for its real time implementation. The algorithm has been substituted with efficient arithmetic modules, converted into parallel algorithm and finally a parallel architecture has been developed.

I. INTRODUCTION

The high resolution direction-of-arrival (DOA) estimation is important in many sensor systems. It is based on the processing of the received signal and extracting the desired parameters of the DOA of plane waves. Many approaches have been used for the purpose of implementing the function required for the DOA estimation and are available in the literature mainly in the various conference proceedings of acoustic speech and signal processing and IEEE Transaction of signal processing [1-6]. The Multiple Signal Classification (MUSIC) and the Estimation of Signal Parameters by Rotational Invariance techniques (ESPRIT) algorithms are widely studied and provide asymptotically unbiased and efficient estimates of the DOA [1-2]. They estimate the so called signal subspace from the array measurements. The parameters of interest (i.e. determining of the DOA) are then estimated from the intersection between the array manifold and the estimated subspace.

An important aspect of the design of a signal processing system for the DOA is the computation of the spectral decomposition. In recent years, the search for useful algorithms and their associated architecture using special purpose processors has been a challenging task [7]. Such high performance processors are often required to be used in real time application; thus, it is felt that they should rely on efficient implementation of the algorithms by exploiting pipeline and parallel processing to achieve a high throughput rate. With the advances in the area of VLSI it is now possible to design special purpose hardware for the implementation of various real time algorithms. The customized hardware has two main advantages as listed below.

1) The given algorithm is executed at a high speed.

2) Cost and size of the hardware will be lower than the cost of a general purpose computer.

These advantages have led many researchers to probe into the

This research is partly supported by ONR Grant # N00014-91-J-1011

possibility of designing special purpose hardware. The development of special purpose hardware will need to exploit pipeline, parallel and distributed processing approaches to achieve high throughput rates. There are many architectures such as systolic array, SIMD Cordic Processors and MIMD which can be used for parallel implementation [7]. An appropriate structure which can exploit maximum parallelization to reduce the computation time will be selected for real time implementation for the particular application.

Although MUSIC is a high resolution algorithm, it has several drawbacks including the fact that complete knowledge of the array manifold is required, and that is computationaly very expensive as it requires a lot of computations to find the intersection between the array manifold and the signal subspace. This drawback can be overcome by using high speed parallel architecture for the computation of DOA. In the following sections first of all MUSIC algorithm is briefly described followed by its parallelization and development of its parallel architecture.

II. MUSIC ALGORITHM

It is assumed that there are m sensors and d narrowband sources. In the discussion of this algorithm an array of eight sensors (m=8) is further assumed. The data vector x(t) is a linear combination of the d steering column vectors of the direction matrix and is therefore constrained to the d-dimensional subspace of Cm termed the signal subspace, that is spanned by the d columns vectors of the direction matrix. In this case the signal subspace intersects the array manifold at the d steering vectors.

However, when the data is corrupted by noise, the signal subspace has to be estimated and consequently it is expected that the signal subspace will not intersect the array manifold, so the steering vectors closest to the signal subspace will be chosen instead [3].In [1-3], it is shown that one set of d independent vectors that span the signal subspace is given by the d eigenvectors corresponding to the d largest eigenvalues of the data covariance matrix. The data covariance matrix is assumed to be positive definite and Hermetian.

Thus the eigenvalues of the data covariance matrix are the d largest eigenvalues of the spatial covariance matrix augmented by noise. Also the (m-d) smallest eigenvalues are all equal to noise.

Thus first d eigen vectors span the same subspace as spanned by the column vectors of direction matrix. In most situations, the covariance matrices are not known exactly but need to be estimated. Therefore, one can expect that there is no intersection between the array manifold and the signal subspace. However, elements of the array manifold closest to the signal subspace should be considered as potential solution. After determining the number of sources [4], Schmith [1] proposed an appropriate function as one possible measure of closeness of an element of the array manifold to the signal subspace. The dominant d peaks are the desired estimates of the directions of arrival.

For the particular case where the array consists of m sensors uniformly spaced, and if the reference point is taken at the first element of the array, Power is obtained by first calculating the DFT of the vectors spanning the null space of eigenvectors. Angles which corresponds to peaks of the power function will be the the angle of arrivals. Summary of the MUSIC algorithm is as follows:

1) Estimate the data covariance matrix R.
2) Perform the eigendecomposition of R.
3) Estimate the number of sources.
4) Evaluate Power function.
5) Find the d largest peaks. of Power to obtain estimates of the parameters.

A. Parallelizing the MUSIC Algorithm

In order to develop a real time parallel architecture for a given algorithm following steps need to be performed. An algorithm should be first converted into a computationally efficient algorithm. In comparing the various methods for solving the eigendecomposition problem, there are numerous factors that one must consider. Perhaps, the primary factor is that of the relative efficiency of the method under consideration. One criterion commonly used in the eigendecomposition problem for determining the efficiency of a particular method is the time required to solve this problem, and hence one might rely on special purpose hardware and exploit pipeline and parallel processing to achieve high throughput rates.

The algorithm is divided into parallel modules. If a particular module of the algorithm can not be executed in parallel due to data dependency it should be pipelined. Degree of parallelizing will also depend on the throughput rate of the system and available time for the computation. Using these guidelines MUSIC algorithm is implemented.

It is clear from the above discussion that the implementation of the MUSIC algorithms requires formation of data covariance matrix and the computation of the eigenvalues. As explained earlier QR algorithm is very promising for the computation of the eigenvalues and the eigenvectors. Since QR algorithm when applied to a dense matrix in this case covariance matrix is very time consuming. One approach to alleviate this problem is first utilize the well known Householder transformation to convert dense matrix into a tridiagonal matrix. Then QR algorithm can be applied to convert tridiagonal matrix into a diagonal matrix giving eigenvalues.

The hardware block diagram for the MUSIC Algorithm is shown in Figure 1. As seen in this figure, the data collected from the sensors is utilized to form the covariance matrix. The eigendecomposition is performed using Householders transformation and QR method. The eigenvalues are used to find the number of sources and finally using the eigenvectors in Power method to find the angle of arrival.

Various papers pertaining to parallelization of Householders and QR algorithms were reviewed. C.F.T. Tang et al [6] and K.J.R. Liu [8] proposed architecture for complex Householder transformations for triangularization of the matrix. In their architecture they used single column with the number of processors equal to the number of columns of the matrix. Each processor performs operation on each column. After each iteration the values of each column are fed back to the same processors. But their architecture is proposed to perform triangularization of the given matrix whereas we are interested in tridiagonalization of the covariance matrix.

QR method for the tridiagonal matrix is also implemented by W. Phillips [9]. In his architecture, rectangular systolic array is used in which each processor performs single iteration. When the first iteration is performed on the m th row by the k th processor, the second iteration is performed on the (m-1) th row by the (k-1) th processor and so on. But the disadvantage in this approach is that the number of processors is dependent on the number of iterations, i.e., if 5 iterations are required then 5 processors are required. But the exact number of iterations is not known which leads to the uncertainty of the required number of processors.

K.J.R.Liu [10] has proposed another kind of approach in which a systolic array arranged in a matrix form is used. The number of processors is equal to the number of elements of the matrix. During first step, the matrix Q is found. Then new values of matrix A are then

calculated using Q. Convergence for all the elements of the matrix other than the diagonal elements is checked. If all the elements of the matrix other than the diagonal elements are not equal to zero then the same systolic array is used for the next iteration. These iterative computations are used until all the elements of the matrix except the diagonal elements converge to zero. The obvious advantage is that the same set of processors can be used for all the iterations. But the drawback is that this architecture is proposed for the evaluation of eigenvalues on the dense matrix.

III. ARCHITECTURE

A. Data Covariance Matrix

Once the data has been collected by the sensors the data covariance matrix can be computed. The sampled data obtained from the sensors is used to obtain the data covariance matrix. Since the covariance matrix is Hermetian, the computation of lower triangular matrix of covariance matrix is sufficient to get complete information of the full matrix.

The parallel computation of the data covariance matrix is performed using systolic architecture. Since there are 36 elements in the lower triangular 8*8 matrix, systolic architecture will have 36 processors. Here a triangular arrangement of the systolic array with global routing is considered as shown in Figure 2. Each processor is numbered as Pmn where m is the row number and n is the column number. The sampled data from the ith sensor is sent to the ith row and the ith column simultaneously. For example the sampled data from the 3rd sensor is sent to all the processors in the third row and the third column. Each processor performs multiplication and addition of two sampled data in parallel in all the processors for every clock cycle . Since there are 36 processors, 36 multiplications and 36 additions are performed simultaneously. Each processor has a memory to store the product of multiplication which is added to the product obtained during the next data cycle. Once the operations of multiplication and addition for all the sampled data in all the processors is performed, the stored data in each processor is then divided by the number of samples in all the processors in parallel. The resulting output are used to form the data covariance matrix.

B. Householder Method

Householders method is a technique used for reducing the bandwidth of the data covariance matrix by transforming it to a tridiagonal one under congruent transformations without affecting the values of the eigenvalues [5].

In order to transform the m*m data covariance matrix to a tridiagonal matrix, T, m-2 Householder's transformations are determined. Each transformation is determined to eliminate a whole row and column above and below the subdiagonals without disturbing any previously zeroed rows and columns. The Householder transformation algorithm has been parallelized and details of the derivation of this algorithm can be found in [11]. A flowchart of this parallel algorithm is given in Figure 3.

As shown in Figure 3, the determination of all the d's and the new elements of the columns of the matrix can be computed in parallel. Thus this algorithm can be mapped on a hardware architecture with the number of processors equal to m+1, where m is the order of the matrix. The architecture is as shown in Figure 4. The columns of the matrix are sent to each processor in a pipelined fashion in reverse order such that the last element of each column becomes the first element. The Processor PE1 is used to find the w and c required by other processors to find the d. Processors PE2, PE3... PE8 are used to find d using the value of w and c found in the first processor. At the same time the first processor is used for the

evaluation of b. The first element of the first column and b are the output of the first iteration which are used as input for evaluation of eigenvalues using QR method. All the d's are evaluated in parallel and are sent to the processor PE9. The processor PE9 is exclusively used for the determination of v using d and w. The v are then routed back to all the processors. The processors PE2, PE3... PE8 use w, d and v to find the new values of the elements of the columns in parallel. The counter is used to set the number of iterations to m-2. For m-2 times, the intermediate results are used in feedback loop and the same set of processors are used repeatedly. The feedback loop has a FIFO memory to temporally store all the elements of the column until operations on previous iteration are completed. For the first iteration, operations on 8*8 matrix are performed hence all the processors are utilized. For the second iteration, operation on 7*7 matrix are performed. Now the first column of the matrix is *already computed*; therefore new elements of the second column from PE2 are fed back to PE. Thus for the second iteration, PE2 does not have any column to work on and is thus disabled. All other processors perform same operation as in the first iteration, but the elements of each column are reduced by one element. Thus for every new iteration the columns and the elements of the columns keeps on reducing.

C. QR Method

Given the tridiagonal matrix T and matrix U which are obtained from Householders transformation, the QR algorithm may be used to compute eigenvalues and eigenvectors. This is achieved by producing a sequence of transformations based on orthogonal matrices. After k iterations T will be approximately a diagonal matrix whose diagonal elements approximate the eigenvalues of the original matrix, and the appropriate eigenvectors are given by the columns of the U matrix. The orthogonal matrices in the QR algorithm are the product of m-1 rotations in the (i,i+1) planes respectively. Each rotation is defined as a matrix which is an identity matrix except for the entries (i,i), (i,i+1), (i+1,i), and (i+1, i+1) which together form a 2*2 matrix. Each subdiagonal element can be eliminated by a plane rotation.

After parallelizing this algorithm whose details are given in [9,11], a parallel architecture has been developed. The architecture uses m+1 processors where 'm' is the number of rows or columns of the tridiagonal matrix. Architecture for 8*8 matrix is shown in Figure 5. Two kinds of processors are used. The processor PE1 is used to compute the eigenvalues and the other eight processors are used to find the eigenvectors of the tridiagonal matrix. The diagonal and subdiagonal elements of the tridiagonal from the Householder's transformation is pipelined into the processor PE1 which performs the first iteration. The new values of rotation are computed and are sent in pipelined fashion to all processors in that column. These processors PE11, PE21....PE81 find the eigenvector of the tridiagonal matrix. The processor PE1 also performs the test for convergence. If the sum of squares of the subdiagonal elements is not nearly equal to zero, PE1 performs the next iteration to find the eigenvalue. This process is repeated until matrix converges to a diagonal matrix. Thus the diagonal elements are the eigenvalues and the every other processor gives the rows of the eigenvector.

D. DOA Estimation

Once the eigenvectors have been computed, the values of the eigenvectors are utilized to calculate the power method. The details of this algorithm can be found in [1,11]. The evaluation of power method first requires squaring of the product of rowvector of the array manifold and the eigenvector matrix. The process of computing the product is repeated for different arrival angles and

they are squared. The squared values are accumulated for all the values in the array manifold. The hardware design to compute the power function or method is shown in Figure 6. It consists of a set of 8 processors. Each processor finds the product of rowvector and the column of eignevectors matrix in parallel. The product obtained is then summed using adders. This sum is squared and added. The power function is thus calculated for different values of the array manifold. This computed value is different for different angle. The angles corresponding to d minimum values (sharp valleys or peaks if the function is inversed) are the angles of arrival.

IV. CONCLUSIONS

First of all MUSIC algorithm has been modified with efficient computational modules. The algorithm has been parallelized. Four modules are implemented using pipeline and parallel processing schemes. The architecture is developed assuming eight sensors and is capable of computing DOA in real time. The architecture is scalable and can incorporate any number of sensors. Real time requirement should be considered at the time of scaling. The architecture utilizes identical processing elements and efforts are directed to develop generelized processing element.

V. REFERENCES:

1. R.O. Schmith,"Multiple emitter location and signal parameter estimation," IEEE Trans. on Antennas and Propagation, Vol AP-34, No.3,PP. 276-280, Mar. 1986.

2. R. Roy and T. Kailath,"ESPRIT-Estimation of signal parameters via rotational invariance techniques," IEEE Trans. Acoustic,Speech and Signal Processing, Vol. 37, No. 7, PP 984-995, July 1989.

3. D. Spielman, A. Paulraj,"Performance analysis of the MUSIC algorithm," in ICASSP 1986., PP. 1909-1912.

4.T. J. Shan and A. Paulraj,"On smoothed rank profile tests in eigenstructure approach to direction-of-arrival estimation," ICASSP 1986, PP 1905-1908.

5.C.Y.Chen and J. A. Abraham,"Fault-tolerant systems for computations of eigenvalues and singular value" SPIE Vol. 696, Advance Algorithm and architecture for signal processing, pp 228-237, 1986.

6. C.F.T. Tang, K.J.R. Liu, S.A. Tretter,"On systolic array for recursive complex Householder transformations with applications to array processing," ICASSP 1991, PP 1033-1036.

7. S.Y. Kung,"VLSI Signal Processors," Prentice Hall 1987.

8.K.J.R. Liu, S.F. Heieh, K. Yao,"Two level pipelined implementation of systolic block Householder transformation," ICASSP 1990, PP. 1631-1634.

9. W. Phillips and W. Robertson,"Systolic architecture for symmetric tridiagonal eigenvalue problem," IEEE International Conference on systolic arrays, PP. 145-150, 1988.

10. K.J.R. Liu, K. Yao,"Multiphase systolic architecture for spectral decomposition," 1990 International conference on parallel processing, PP I-123-I-126.

11.M. M. Jamali, S.C. Kwatra,"Development of parallel architectures for sensor array processing algorithms."Report No. DSPH-2, University of Toledo, 1992.

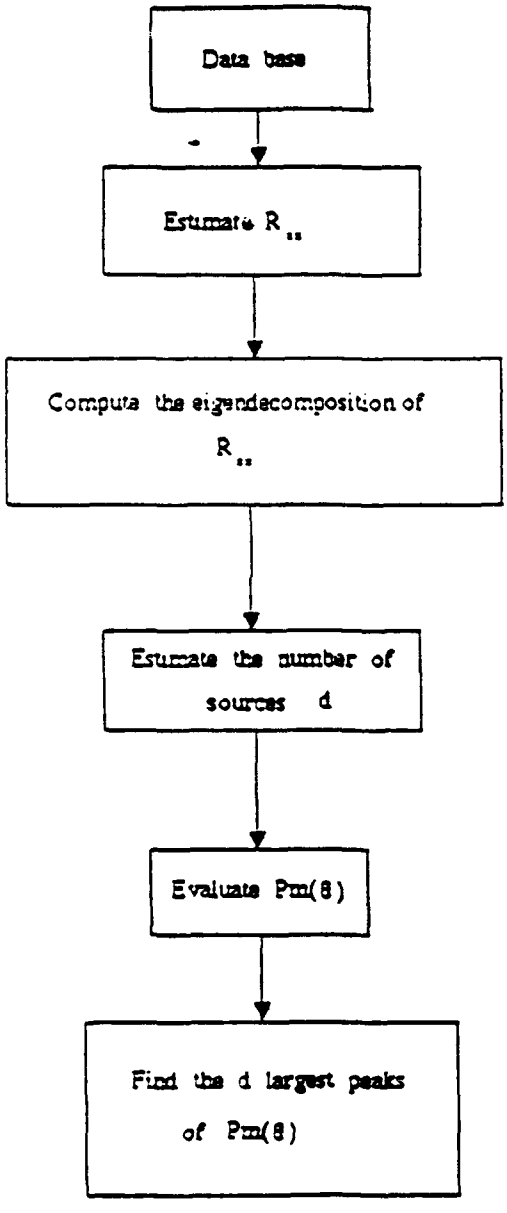


Figure 1: Flowchart for MUSIC Algorithm

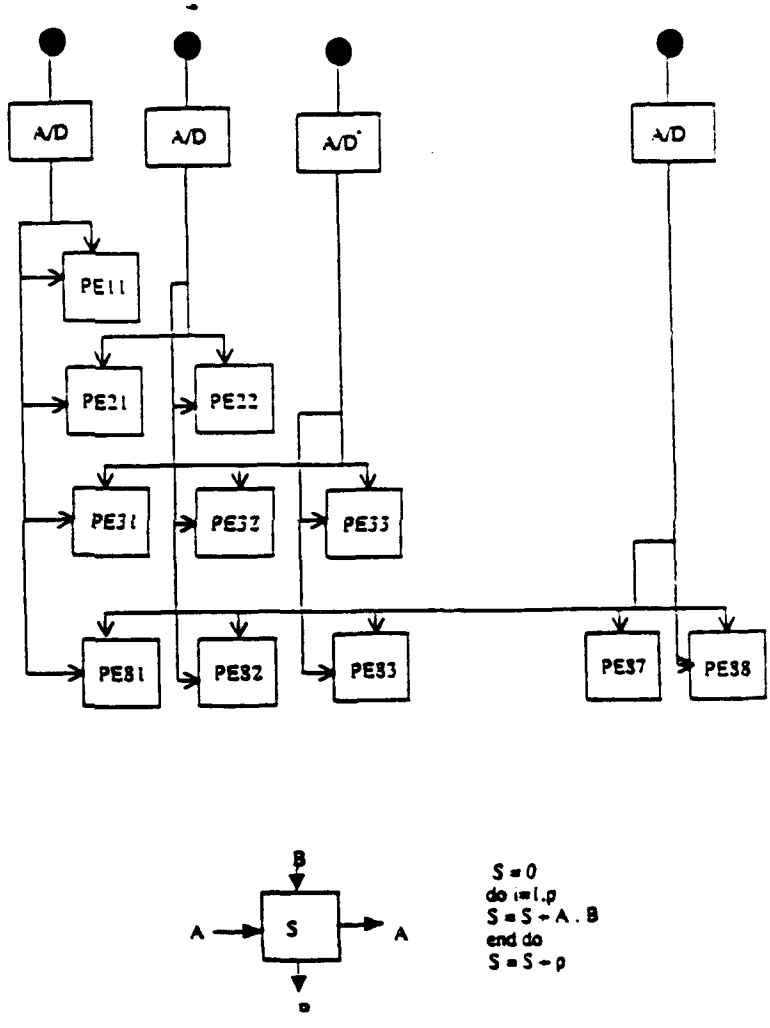


Figure 2: Architecture for Computation of Covariance Matrix.

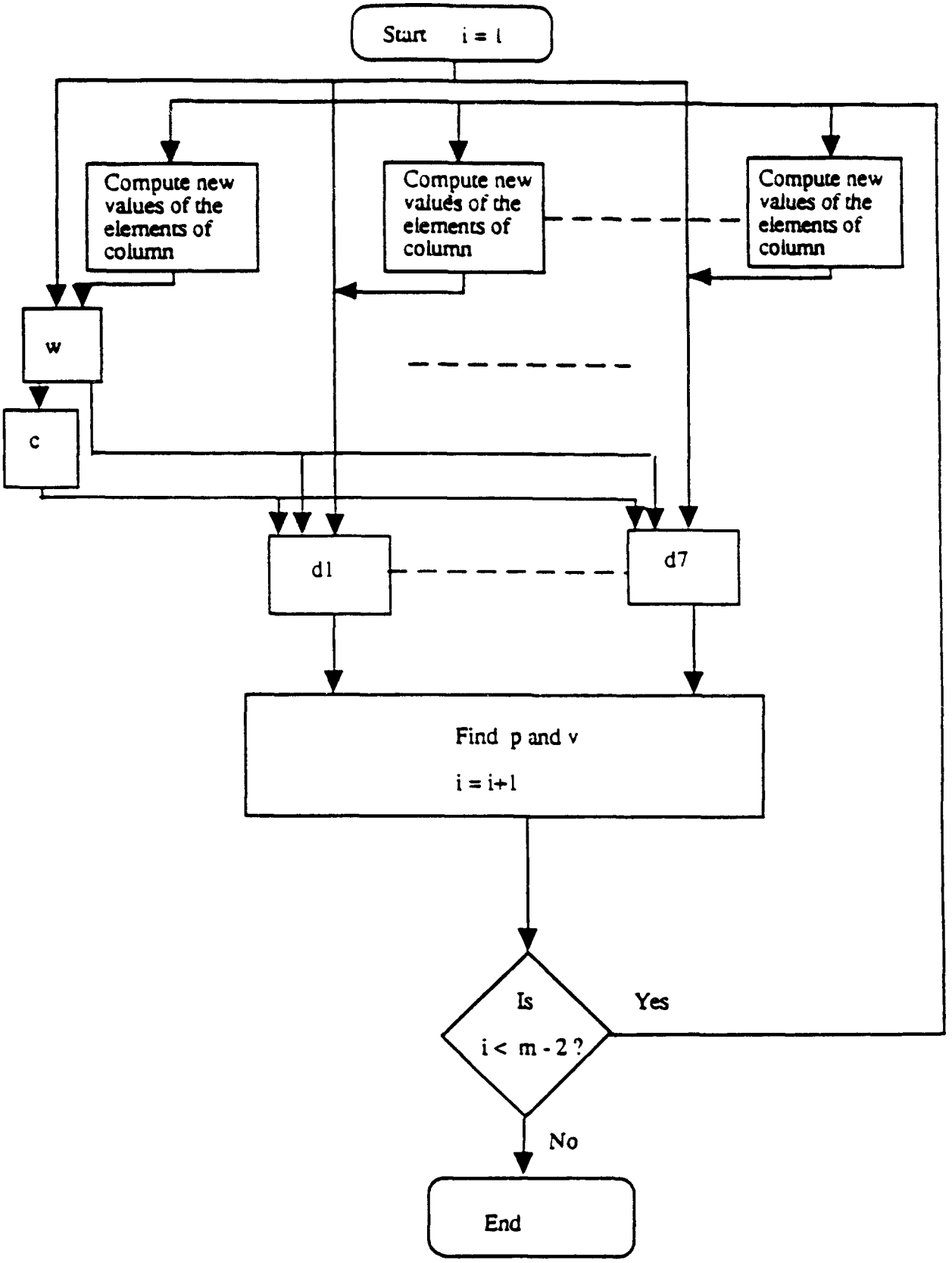


Figure 3:Flowchart for Parallel Householders transformation

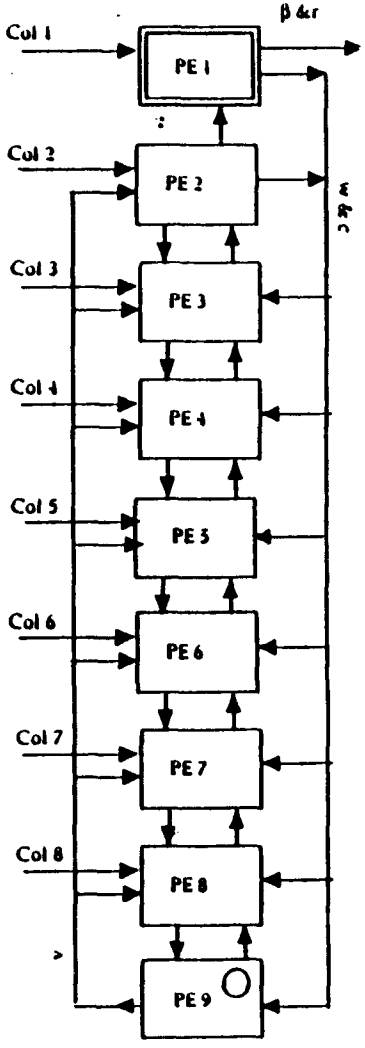


Figure 4: Architecture for Householder Transformation

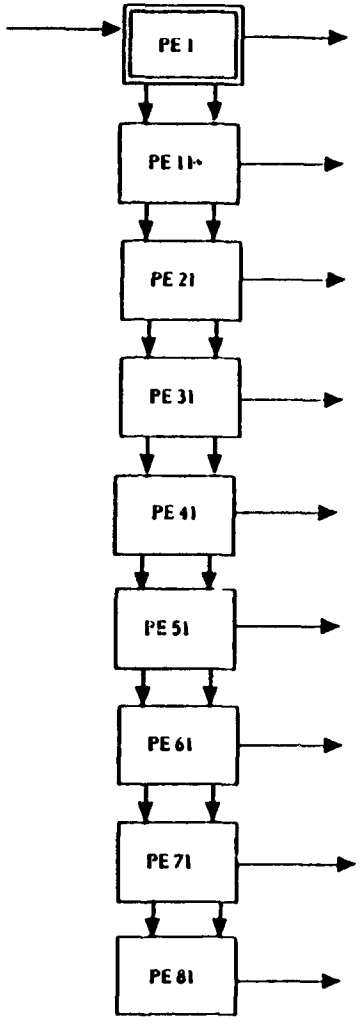


Figure 5: Architecture for QR Method

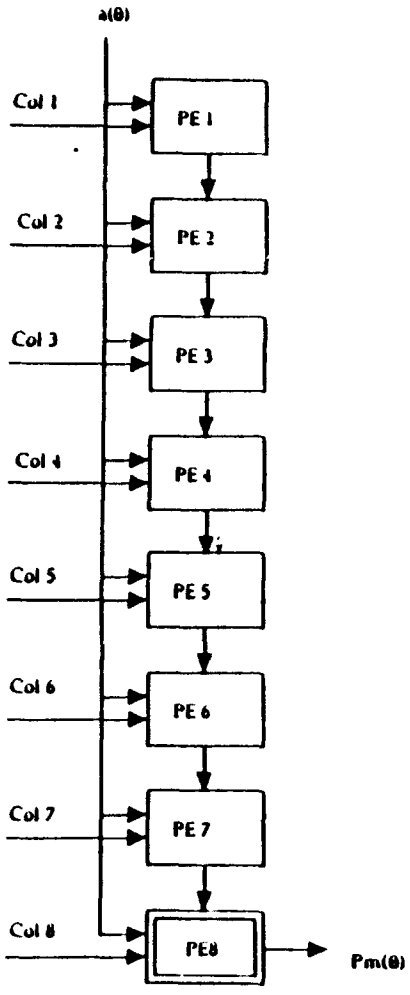


Figure 6: Architecture for Power Method

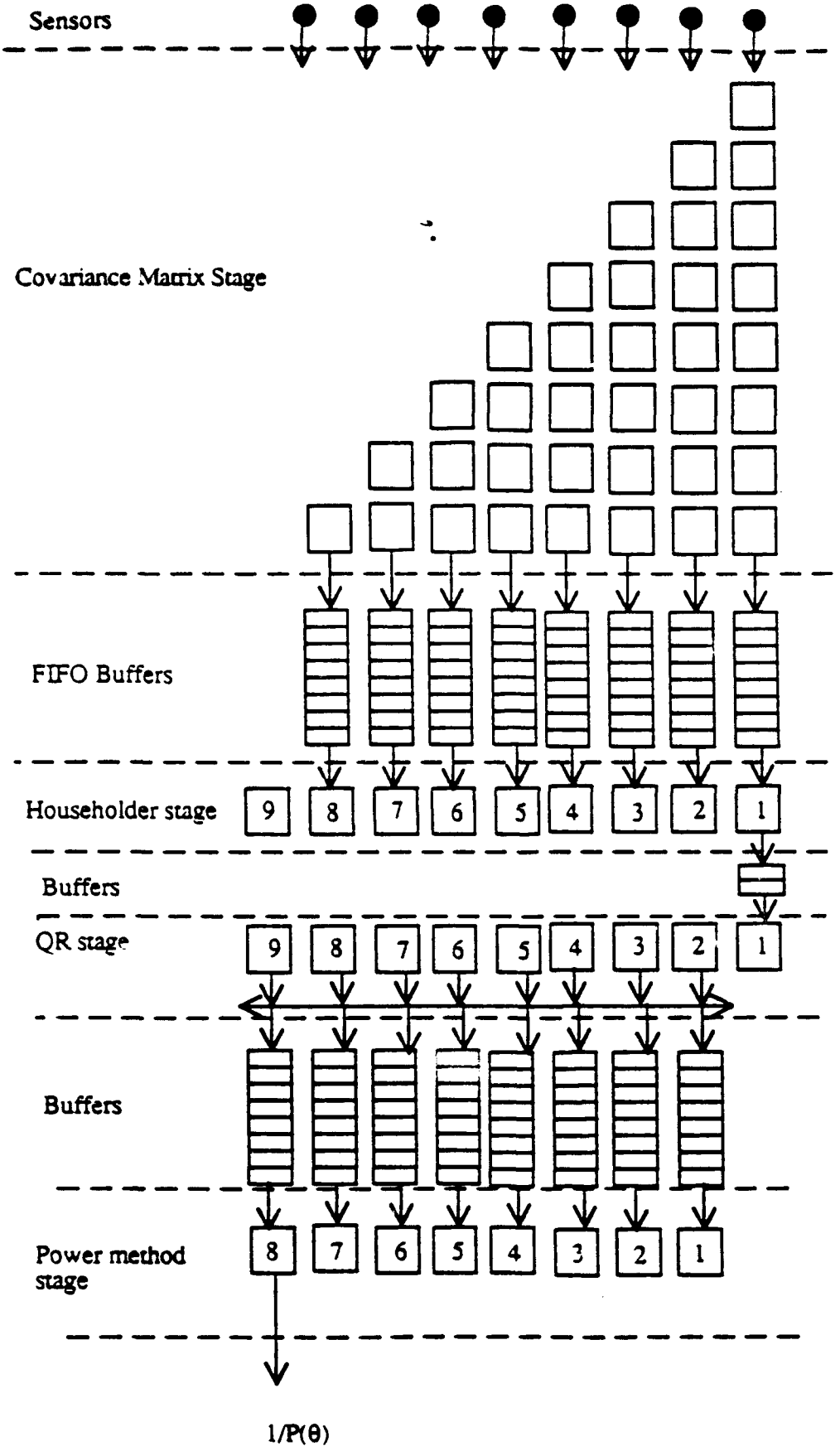


Figure 7:Architecture for MUSIC Algorithm

Conference Record

Twenty-Sixth Asilomar Conference on Signals, Systems & Computers

OCTOBER 26–28, 1992
PACIFIC GROVE, CALIFORNIA

Volume 1 of 2

IEEE Computer Society Press

The Institute of Electrical and Electronics Engineers, Inc.

A PARALLEL QR ALGORITHM FOR THE SYMMETRICAL TRIDIAGONAL EIGENVALUE PROBLEM

A. Djouadi, M. M. Jamali, S. C. Kwatra
Department of Electrical Engineering
University of Toledo

ABSTRACT

In this paper, we propose a parallel/pipelined algorithm and its architecture to solve the symmetric eigenvalue problem. This algorithm is based on Given's rotations, and it is associated with the initial reduction of the dense matrix to a tridiagonal one using Householder's transformations. The performance of this algorithm is described and compared to the performance of the sequential one. It can be shown that the cost of the eigendecomposition falls from O(mxn) to O(m+n) , where m and n denote the matrix order and the number of iteratons respectively. This algorithm is mapped on m processors to compute the eigenvalues and eigenvectors simultaneously.

INTRODUCTION

An important aspect of the design of many modern signal processing systems is the computation of the spectral decomposition. In recent years, the search for useful algorithms and their associated architecture using special purpose processors has been a challenging task. Such high performance processors are often required to be used in real time application; thus, it is felt that they should rely on efficient implementation of the algorithms by exploiting pipelining and parallel processing to achieve high throughput rate. The QR algorithm is one of the most promising suitable for the spectral decomposition problem due to its stability, convergence rate properties, and suitability for VLSI implementation [1]. A useful property of the QR transformations is that shifts can be used to increase the rate of convergence to locate the eigenvalues [2], [3]. This may be very useful for some systems applications where the computations of the eigenvalues are sufficient, such as matrix rank determination and system identification [1]. However, in other applications, (e.g., direction of arrival estimation, spectral estimation, and antenna beamformation), the computation of both the eigenvectors and eigenvalues is crucial [1], and one might use the QR algorithm without shifts to obtain these parameters in parallel. In such case, this algorithm may require a sufficiently large number of iterations to converge. Keeping a low number of iterations may yield to inferior results such as in MUSIC and ESPRIT algorithms, where an accurate computation of the eigenvalues and eigenvectors will also determine the accuracy of the direction of arrival (DOA's) [6] . Also, one drawback of the QR algorithm is that when applied to a dense matrix, it may bevery time-consuming and may pose difficulties for parallel implementation due to communication and timing among different modules of the systolic array [1]. Hence, it is generally not feasible to carry out the QR transformations on a dense matrix. Instead, if the dense is first reduced to a tridiagonal matrix T, using Householder's transformations, the cost of the eigendecomposition falls substantially from $O(m^3)$ to O(m) [2], where m denotes the matrix order. Furthermore, in [4] a sequential algorithm was proposed for solving the symmetric tridiagonal eigenvalue problem. Although this algorithm reduces the processing time at some extent by avoiding the computation of the product RQ in the QR algorithm [2] at every iteration, and also the storage of the matrix R, but still every iteration in the algorithm requires m steps. For n iterations mxn steps are required to perform the eigendecomposition of the tridiagonal matrix.

In this paper, we present a parallel/pipelined algorithm for the tridiagonal symmetrical eigendecomposition problem. This algorithm is capable of generating the eigenvalues and eigenvectors simultaneously by pipelining the successive iterations readily.

THE SYMMETRIC EIGENDECOMPOSITION PROBLEM

It is well known that the symmetric eigendecomposition of the data covariance matrix is one of the most fundamental problem in signal processing as it arises in many applications such as DOA's estimation and spectral estimation. Householder's method is a technique used for reducing the bandwidth of the data covariance matrix by transforming it to a tridiagonal one under congruent transformations without affecting the values of the eigenvalues, and thus reducing subtancially the cost of the eigendecomposition [2], [5]. In order to transform the mxm data covariance matrix R_{xx} to a tridiagonal one, (m-2) Householder' s transformations are determined such that

This research is partly supported by NAVY grant N00014-91-J-1011

Presented at Asilomar Conference on Signals, Systems & Computers
October 26-28, 1992 Pacific Grove, CA

1058-6393/92 $03.00 © 1992 IEEE

$$N^H R_{xx} N = T \quad (1)$$

where

$$N = N_1 . N_2 \ldots N_{m-2} \quad (2)$$

Each transformation is determined to eliminate a whole row and column above and below the subdiagonals without disturbing any previously zeroed rows and columns. Given the tridiagonal matrix T and the Householder's transformation matrix N, the QR algorithm may be used to compute the eigenvalues and the eigenvectors simultaneously. This is achieved by producing a sequence of transformations based on orthogonal matrices and illustrated by the following algorithm.

$$T_1 = T \quad (3)$$

$$U_1 = N^H \quad (4)$$

begin

for i=1, n

$$R_i = Q_i^H T_i \quad (5)$$

$$T_{i+1} = R_i Q_i \quad (6)$$

$$U_{i+1} = Q_i^H U_i \quad (7)$$

endfor

$$\Sigma = T_{n+1} \quad (8)$$

$$X = U_{n+1} \quad (9)$$

After k iterations T will be approximately a diagonal matrix Σ whose diagonal elements approximate the eigenvalues of the original matrix, and the appropriate eigenvectors are given by the rows of the matrix X. The orthogonal matrices Q_i in the QR algorithm are the product of m-1 rotations $Q_{(i,j)}$; j=1,...,m-1, in the (j,j+1) planes respectively. Each rotation $Q^H_{(i,j)}$ is defined as a matrix which is an identity matrix except for the entries (j,j), (j,j+1), (j+1,j), and (j+1, j+1) which together form a 2 x 2 matrix given by

$$Q^H_{(k,i)} = \begin{bmatrix} c & s^* \\ -s & c \end{bmatrix} \quad (10)$$

The factorization producing R_i and Q_i from the original matrix T is explained as follows, we simply eliminate each subdiagonal element by a plane rotation, the first one is

$$Q^H_{(1,1)} R_1 =$$

$$\begin{bmatrix} 1 & \exp(-j\theta) \\ -\exp(j\theta) & 1 \end{bmatrix} \begin{bmatrix} t_{11} & \cdot \\ t_{21} & \cdot \end{bmatrix} \quad (11)$$

The (2,1) entry in this product should be equal to zero

$$-t_{11} \exp(j\theta) + t_{21} = 0 \quad (12)$$

$$\exp(j\theta) = t_{21}/t_{11} = r \exp(j\phi) \quad (13)$$

where $r = |t_{21}/t_{11}|$, and $\phi = \arg(t_{21}) - \arg(t_{11})$

To have a unitary matrix, the matrix $Q^H_{(1,1)}$ is chosen as

$$\begin{bmatrix} \frac{1}{\sqrt{1+r^2}} & \frac{r\exp(-j\phi)}{\sqrt{1+r^2}} \\ \frac{-r\exp(j\phi)}{\sqrt{1+r^2}} & \frac{1}{\sqrt{1+r^2}} \end{bmatrix} \quad (14)$$

For a Hermetian tridiagonal matrix, we have

$$\frac{1}{\sqrt{1+r^2}} = \frac{t_{11}}{\mathrm{sqrt}(t_{11}^2 + |t_{21}|^2)} \quad (15)$$

and

$$\frac{r\exp(j\phi)}{\sqrt{1+r^2}} = \frac{t_{21}}{\mathrm{sqrt}(t_{11}^2 + |t_{21}|^2)} \quad (16)$$

Let the entries of the diagonal and lower subdiagonal of the tridiagonal matrix T be a(j,i) and b(j,i) respectively where j is the rowor column number and i is the iteration number, c(j,i+1) and s(j,i+1) be the elements of the rotations used in rotating rows j and (j+1) at the (i+1)th iteration, and x(j, i+1) the diagonal element of T after rotation of rows j and j+1 at the (i+1)th iteration. Also, let the entries of the eigenvectors be u(j,l,i), where j and l are the row and column number, then a pseudecode of an updated version of the sequential algorithm given in [4] to compute the eigenvalues and eigenvectors is as follows :

b(1, .)=0; a(m+1, .)=0; b(m+1, .)=0; c(0, .)=1;
s(0, .) = 0; c(m+1, .)=1; s(m+1, .)=0;
for i= 1, n
x(0, i+1)=a(1, i);

```
for j=1,m
  if (x(j-1, i+1) > 0
    r=sqrt{|b(j+1, i)|^2 +x(j-1, i+1)^2 }
    c(j, i+1) = x(j-1, i+1)/r
    s((j, i+1) = b(j+1, i+1)/r
  else
    c(j, i+1) = 1
    s(j, i+1) = 0
  end if
  w=c(j, i+1).x(j-1, i+1)+s*(j, i+1).b(j+1, i)
  v=c(j-1, i+1).c(j, i+1).b*(j+1, i)
    +s*(j, i+1). a(j+1, i)
  b(j, i+1)=s(j-1, i+1) . w
  a(j, i+1)=c(j-1, i+1).c(j, i+1).w + s(j, i+1).v
  x(j, i+1)=c(j, i+1) a(j+1, i)
            -c(j-1, i+1) s( j, i+1) b(j+1, i)

  for l=1,m
  u(j, l, i+1)=c(j, i+1).u(j, l, i)
                +s*(j, i+1).u(j+1, l, i)
  u(j+1, l, i+1)=-s(j, i+1).u(j, l, i)
                  +c(j, i+1).u(j+1, l, i)
  end for
 end for
end for
```

PARALLEL ALGORITHM FOR THE TRIDIAGONAL QR ALGORITHM

An attempt is made here to parallelize the previous sequential algorithm to reduce the number of steps from mxn to 2n+m-2 steps. A parallel/ pipelined algorithm has been developed and is described in terms of a simple program given below. The program consists of odd and even steps; during an odd step, the odd terms (a(i,.), b(i,.), i=1,3,...m-1, of the matrix T are updated in parallel. Likewise, during an even step, the even terms a(i,.), b(i,.), i=2,4...,m, are updated in a similar fashion.

```
b(1, . )=0; a(m+1, . )=0; b(m+1, . )=0; c(0, . )=1;
s(0, . )=0; c(m+1, . )=1; s(m+1, . )=0;
n = number of iterations
for k= 1,n+(m-2)/2
Odd Steps
  for j=1,m-1,2
  i=k-(j+1)/2
    Parallel
      if (i.ge.0. and i.le. (n-1)) then
        x(0, i+1)=a(1, i)
        if (x(j-1, i+1).eq. 0) then
          c(j, i+1)=1
          s(j, i+1)=0
        else
          r=sqrt(|b(j+1, i)|^2+x(j-1, i+1)^2)
          c(j,i+1)=x(j-1, i+1)/r
          s(j,i+1)=b(j+1, i)/r
        endif
        w=c(j, i+1).x(j-1, i+1)
          +s*(j, i+1) . b(j+1, i)
        v=c(j-1, i+1) . c(j, i+1) . b* (j+1, i)
          + s* (j, i+1) . a(j+1, i)
        b(j, i+1)=s(j-1, i+1) . w
        a(j, i+1)=c(j-1, i+1).c(j, i+1).w
                  + s(j, i+1).v
        x(j, i+1)=c(j, i+1)a(j+1, i)
                  -c(j-1, i+1)s( j, i+1) b(j+1,i)

        for l=1, m
          u(j, l, .) = c(j, i+1).u(j, l, .)
                       +s*(j, i+1) . u(j+1, l, .)
          u(j+1, l, .) = -s(j, i+1).u(j, l, .)
                         +c(j, i+1) . u(j+1, l, .)
        end for
      endif
    End parallel
  endfor

Even steps
  for j=2,m,2
  i=k - j/2
    Parallel

      Same as above

    End parallel
  endfor
endfor
```

An example of this algorithm applied to a matrix of order 8, and for 11 iterations is shown in Table. 1, where the pairs (i ,j) were defined earlier.

This algorithm is also suitable for VLSI implementation, using an array of m/2 processors Pr_1, Pr_2,..., $Pr_{m/2}$ and (m+2) cells cl $_0$, cl $_1$,..., cl $_{m+1}$ consisting a local memory, as shown in Figure 1. Each processor in the array performs certain computations such as floating point operations and square roots.

If the pairs (a(j,), b(j,.)) , j=1,2,....,m . are stored respectively in cl $_1$,..., cl$_m$., (c(0,.) , s(0, .)) are stored in cl $_0$, and (a(m+1,.) , b(m+1,.)) are stored in cl $_{m+1}$, then during an odd step , each processor Pr_j, respectively

1) accepts
   a) c(2j-2, i+1), s(2j-2, i+1), and x (2j-2, i+1) from cell cl $_{2j-2}$,
   b) a(2j,i), b(2j,i) from cell cl$_{2j}$,
2) compute x(2j-1,i+1), c(2j-1,i+1), and s(2j-1,i+1),
3) updates a(2j-1,i) and b(2j-1,i) to become a(2j-1, i+1) and b(2j-1, i+1) respectively,
4) store x(2j-1,i+1), c(2j-1,i+1), s(2j-1,i+1), a(2j-1,i+1), and b(2j-1, i+1) in cell cl $_{2j-1}$.

Step	Compuations performed in parallel
1	(1,1)
2	(2,1)
3	(1,2) (3,1)
4	(2,2) (4,1)
5	(1,3) (3,2) (5,1)
6	(2,3) (4,2) (6,1)
7	(1,4) (3,3) (5,2) (7,1)
8	(2 4) (4,3) (6,2) (8,1)
9	(1,5) (3,4) (5,3) (7,2)
10	(2,5) (4,4) (6,3) (8,2)
11	(1,6) (3,5) (5,4) (7,3)
12	(2,6) (4,5) (6,4) (8,3)
13	(1,7) (3,6) (5,5) (7,4)
14	(2,7) (4,6) (6,5) (8,4)
15	(1,8) (3,7) (5,6) (7,5)
16	(2,8) (4,7) (6,6) (8,5)
17	(1,9) (3,8) (5,7) (7,6)
18	(2,9) (4,8) (6,7) (8,6)
19	(1,10) (3,9) (5,8) (7,7)
20	(2,10 (4,9) (6,8) (8,7)
21	(1,11) (3,10) (5,9) (7,8)
22	(2,11) (4,10) (6,9) (8,8)
23	(3,11) (5,10) (7,9)
24	(4,11) (6,10) (8,9)
25	(5,11) (7,10)
26	(6,11) (8,10)
27	(7,11)
28	(8,11)

Table 1 . Example of the parallel/pipelined algorithm for updating the entries a's and b's of a tridiagonal matrix (matrix order = 8,number of iteration = 11)

and during an even step , each processor Pr_j, respectively

1) accepts
   a) c(2j-1, i+1), s(2j-1, i+1), and x (2j-1, i+1) from cell cl $_{2j-1}$,
   b) a(2j+1,i), b(2j+1,i) from cell cl $_{2j+1}$,
2) compute x(2j,i+1), c(2j,i+1), and s(2j,i+1),
3) updates a(2j,i) and b(2j,i) to become a(2j, i+1) and b(2j, i+1) respectively,
4) store x(2j,i+1), c(2j,i+1), s(2j,i+1), a(2j,i+1), and b(2j, i+1) in cell cl $_{2j}$.

The previous array , shown in Fig. 1, can be extended to include another m/2 processors P_1, P_2,.....,$P_{m/2}$, as shown in Fig. 2 to update the matrix of the eigenvectors. Given the matrix $U_1 = N^H$, obtained from Householder's transformations, and the matrix $Q = Q_1 Q_2 \ldots Q_n$, where n is the number of iterations. The product of Q^H by U_1, to obtain the matrix of eigenvectors of the original problem, may be computed also in (2n+m-2) steps.

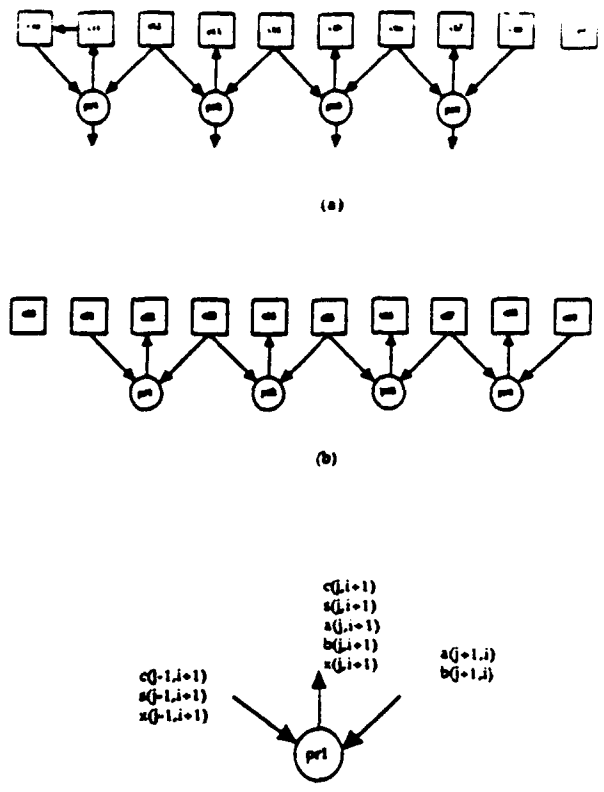


Fig.1. Updating the eigenvalues, (a)odd steps, (b)even steps

If each column of the matrix $U_1 = N^H$ is stored in an array of m elements consisting of a FIFO as depicted in Fig.2, then during an odd step, the values stored at the top of the independent pairs of arrays (1,2), (3,4),...., (m-1, m) are transfered in parallel to the processors P_1, P_2 , ...,$P_{m/2}$ respectively. The rotation parameters generated during this particular step are also sent to the corresponding processors. that is , the rotation parameters generated by Pr_1 are sent to P_1 , and the rotation parameters generated by Pr_2 are sent to P_2, and so on. Once the top elements are updated , they are transferred to the buttom of the corresponding arrays The procedure continues until all the elements stored in the array are updated. This is depicted in Figure 2 (a). Likewise, during an even, updating the entries of the independent column pairs (2,3), (4,5),...., (m-2, m-1) is shown in Figure 2 (b).

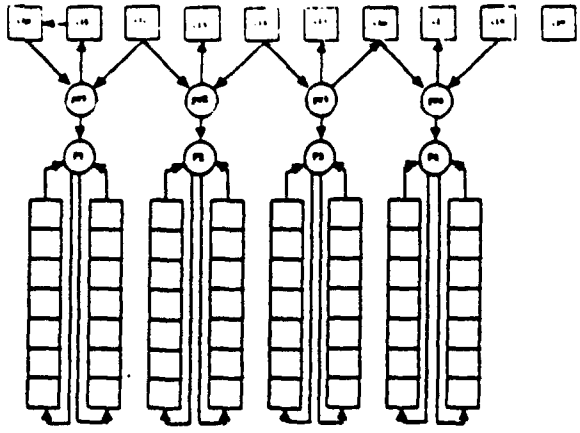


Fig 2 (a). Updating the eigenvectors during an odd step

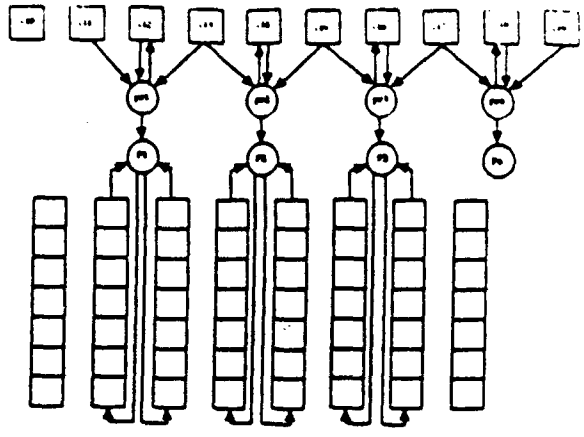
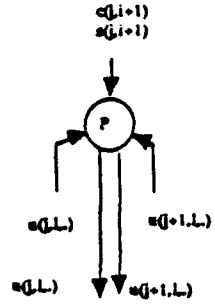


Fig 2 (b). Updating the eigenvectors during an even step.



CONCLUSION

The pipelined/parallel algorithm proposed in this paper can be used efficiently to solve the symmetrical eigenvalue problem for both real and complex cases. However, this algorithm is associated with the initial reduction of the dense matrix to a tridiagonal one using Householder's transformations. The performance of this algorithm was described and compared to the performance of the sequential one. It was shown that this algorithm outperforms the sequential one as only 2n+m-2 steps are needed to perform the eigendecomposition of a tridiagonal matrix. as compared to mxn steps using the sequential one, where m and n denote the matrix order and the number of iteraton respectively. This algorithm was also mapped on a parallel architecture suitable for array processing elememts.

REFERENCES

[1] K. R, Liu and K. Yao, " Multiphase Systolic Algorithms for Spectral Decomposition, " IEEE Transactions on Signal Processing, vol. 40, pp. 190- 201, 1992

[2] J. H. Wilkinson, "The algebriac eigenvalue problem," Clarendon Press, Oxford, Chapter 4, 1965.

[3] G. H. Golub and C. F. Van Loan, *Matrix Computation*, 2nd ed. Baltimore, MD, Johns Hopkins Press, 1989.

[4] W.Phillips and W.Robertson,"Systolic architecture for symmetric tridiagonal eigenvalue problem", IEEE International conference on systolic arrays, pp. 145-150, 1988.

[5] C. F. T , Tang and K. J. R. Liu, " VLSI algorithm for complex householder transformation with application to array processing", Journal of VLSI signal processing , 4, pp. 53-68, 1992.

[6] R. Roy and T. Kailath, " Esprit-Estimation of Signal Parameters via Rotational Invariance Techniqu, ", in Proc. IEEE trans. of ASSP, Vol. 37, pp. 984-995, 1989

MODELING AND SIMULATION
VOLUME 23

Part 3

Computers, Computer Architecture, Microprocessors in Education, Transportation, Optimization, Education

Editors

William G. Vogt
Marlin H. Mickle

Proceedings of the Twenty-Third Annual Pittsburgh Conference
Held April 30 - May 1, 1992
School of Engineering - University of Pittsburgh

Published and Distributed by
School of Engineering - University of Pittsburgh

HOUSEHOLDERS ALGORITHM ON HYPERCUBE ARCHITECTURE

D. Kaur, R. B. Sheelvant, A. H. Djoudi, M. M. Jamali, S. C. Kwatra

Department of Electrical Engineering
University of Toledo
Toledo, OH 43606

ABSTRACT

To achieve computational efficiency, Householders transformation is known to be one of the best orthogonal factorizations technique for the matrices. It is also known that Householders transformation outperforms the Givens rotation in numerical stability under finite-precision implementation, and that it requires fewer arithmetic operations than modified Gram-Schimidt does. Hence in this paper a hypercube architecture for tridiagonalization of symmetric matrix using Householder's method is proposed. The purpose is to speed up the reduction of dense matrix into a tridiagonal matrix by performing various iterations in parallel and by using broadcasting feature of the hypercube architecture.
Key words: Hypercube, Householders algorithm, Parallel processing, broadcasting.

I. INTRODUCTION

With recent advances in the area of VLSI it is possible to design special purpose hardware in real time signal processing for the computation of high resolution direction-of-arrival (DOA). The algorithms of recent interest in computation of high resolution direction-of-arrival (DOA) estimation are Multiple Signal Classification (MUSIC) by Schimdt [1] and Estimation of Signal Parameter via Rotational Invariance Technique (ESPRIT) by Roy [2]. Many other researchers have also shown interest in improvising MUSIC and ESPRIT [3],[4]. All these algorithms use eigenvalues and eigenvectors for the computation of DOA. Hence fast and accurate evaluation of eigenvalues and eigenvectors are very important for the real time hardware implementation of these algorithms. One method is to use QR algorithm to reduce the matrix to diagonal matrix and the elements of the diagonal, are the eigenvalues of that matrix. However when QR method is performed on the dense matrix, the process becomes very time consuming, requiring a large number of computations. In order to circumvent this drawback, the matrix is transformed first to tridiagonal one using Householders transformation. Chen [5], Dongarra [6] and Phillips[7] discussed the reduction of a symmetrical matrix to a tridiagonal matrix using Householders transformation [8]. Hence QR decomposition using Householders transformation is very promising for VLSI implementation of real-time high throughput modem signal processing.

In section II mathematical aspect of Householder's method are presented, then the number of iterations which can be performed in parallel is inspected. In section III pseudocode is written to indicate how this algorithm performs on the hypercube. Computer simulation of the algorithm is done to counter check the validity of the parallel algorithm. In section IV comparison between speed up for hypercube architecture and for systolic architecture is done.

This research is partly supported by NAVY grant N00014 - 91 - J - 1011 and is part of M. S. thesis requirement of Mr. R. B. Sheelvant.

II. HOUSEHOLDERS ALGORITHM

The method of Householder [8][9], can effectively reduce the bandwidth of the matrix R by transforming it to a tridiagonal matrix T. In order to transform the m × m data matrix into a tridiagonal one, m-2, N Householder's transformations are determined such that

$$N^T R N = T$$

where $N = N_{m-2} N_{m-3} \cdots N_1$

$$N_k = \begin{bmatrix} I & 0 \\ 0 & \bar{N}_k \end{bmatrix} \begin{matrix} k \\ m-k \end{matrix} \qquad \begin{matrix} k & m-k \end{matrix} \qquad \text{(i)}$$

and

$$\bar{N}_k = I - \frac{2 w w^T}{w^T w} \qquad \text{with } N_k \in R^{m \times m} \text{ and } w \in R^m \qquad \text{(ii)}$$

The matrices N_k are determined to eliminate the elements above and below the subdiagonals without disturbing any previously zeroed rows and columns. The basic iterative sequence of operations for this transformation method can be stated as

$R_1 = R$

begin

For k=1,2.....m-2,

$$R_{k+1} = N_k^T R_k N_k \qquad \text{such that } N_k^T N_k = I$$

end

$T = R_{m-2}$

For k=1, the transformation can be written as

$$R_2 = N_1^T R_1 N_1 \qquad \text{(iii)}$$

where

$$R_1 = \begin{bmatrix} r_{11} & r_1^T \\ r_1 & \bar{R}_1 \end{bmatrix} \begin{matrix} 1 \\ m-1 \end{matrix} \qquad \begin{matrix} 1 & m-1 \end{matrix} \qquad \text{(iv)}$$

Therefore substituting (iv) in (iii):

$$R_2 = \begin{bmatrix} 1 & 0 \\ 0 & \bar{N}_1^T \end{bmatrix} \begin{bmatrix} r_{11} & r_1^T \\ r_1 & \bar{R}_1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \bar{N}_1 \end{bmatrix}$$

$$R_2 = \begin{bmatrix} r_{11} & r_1^T \bar{N}_1 \\ \bar{N}_1^T r_1 & \bar{N}_1^T \bar{R}_1 \bar{N}_1 \end{bmatrix} \begin{matrix} 1 \\ m-1 \end{matrix} \qquad \begin{matrix} 1 & m-1 \end{matrix}$$

In order for the first term $\bar{N}_1^T r_1$ to be null except for the first element, 'w' should have the following form

$w = r_1 + \beta e_1$

where $e_1 = (1, 0, 0, 0, \ldots)^T$

$$r_1 = (a_{21}, a_{31}, \ldots\ldots a_{n1})^T$$

and $\beta^2 = r_1^T r_1$

The column is given by $\overline{N}_1^T r_1$. Substituting the value of $(N)_1^T$ as in equation (ii) we get

$$\overline{N}_1^T r_1 = r_1 - \frac{2(r_1 + \beta e_1)(r_1^T + \beta e_1^T) r_1}{(r_1^T + \beta e_1^T)(r_1 + \beta e_1)}$$

Thus first column has the form $[a_{11}, -\beta, 0, 0, \ldots\ldots 0]^T$, and because of symmetry so will the first row. Therefore in one step first row and first column is reduced. This is the basis of Householder's method. Expanding (iii) by substituting (ii)

$$R_2 = N_1 R_1 N_1$$

$$= \left[I - \frac{2ww^T}{w^T w} \right] R_1 \left[I - \frac{2ww^T}{w^T w} \right]$$

$$= R_1 - \frac{2ww^T}{w^T w} R_1 - R_1 \frac{2ww^T}{w^T w} + 4 \frac{ww^T R_1 ww^T}{w^T w w^T w} \qquad \text{(v)}$$

Let

$$c = \frac{w^T w}{2}$$

and,

$$d = R_1 w$$

Equation (v) can be rewritten as follows:

$$R_1 = R_1 - \frac{w d^T}{c} - \frac{d w^T}{c} + \frac{w (w^T d) w^T}{c^2}$$

$$= R_1 - \left(\frac{d}{c} - \frac{w (w^T d)}{2 c^2} \right) w^T - w \left(\frac{d^T - (w^T d) w^T}{c - 2 c^2} \right) \qquad \text{(vi)}$$

defining

$$v^T = \frac{d^T - d^T w w^T}{c - 2 c^2}$$

Therefore

$$v = \frac{d}{c} - \frac{w (w^T d)}{2c^2}$$

Substituting v , equation (vi) becomes

$$R_2 = R_1 - w v^T - v w^T \qquad \text{(vii)}$$

The choice of above equation is primarily motivated by the interest in the application of parallel processing in computing the elements of the matrix R_k, that is, all the columns of R_k can be computed in parallel as:

$$R_{k\,j} = R_{k-1\,j} - v_j w - w_j v$$

$j = 1, 2, \ldots\ldots n$
$k = 1, 2, \ldots n-2$

where $R_{k\,j}$ and $R_{k-1\,j}$ are the j^{th} column of R_k and R_{k-1}, and v_j and w_j are the j^{th} components of v and w respectively.

III. MAPPING OF HOUSEHOLDER'S METHOD ON HYPERCUBE

If $J = 2^l$ is the order of matrix then it can be mapped on l dimension hypercube with n host processor. Each node of hypercube has a processor. All the processors except the J^{th} processor in the hypercube have same architecture. The pseudo-code to perform Householder's method on the hypercube is as below. The process is repeated for n-2 times

```
1     i=0
2           s=0
3           do j=n-i+1, -1 in the host processor
4                 w(j-1) = a(j)
5                 s=s+a(j)*a(j)
6           end do
7           s=s+(a(n-2))*(a(n-2))
```

```
8       β = sqrt(s)
9       w(i+1) = r(i+2) + β
10      c = s + r(i+2) * β
11      broadcast w to all the nodes i to m
12      broadcast c to all the nodes i to m
13      do k = i, m-1
14            d(k)=0
15            do j = m, i+2, -1 in nodes i to m-1 in parallel
16                  d(k) = d(k) + r(j) * w(j-1)
17            end do
18            d(k) = d(k)/c in nodes i to m-1 in parallel
19      end do
20      broadcast d from i ... m-1 to m
21      p = 0 in node m
22      do j = m, i+2, -1
23            p = p + w(j-1)*d(j-1)
24      end do
25      p = p/2*(c*c)
26      do j = m, i+2, -1
27            v(j-1) = d(j-1) - w(j-1)*p
28      end do
29      broadcast v to all nodes i to m-1
30      do j=m, i+2, -1 in nodes i to m-1 in parallel
31            do k=m, i+2, -1
32                  r(j,k) = r (j,k) - w(k)*v(j) - v(k)*w(j)
33            end do
34      end do
35      do j=i+2,m
36            transfer r(j) from node i to the host processor
37      end do
38  i = i+1
39  if i ≤ m-2 goto 2
40  end
```

Figure 1 shows matrix of order J=8 (i.e 8×8 matrix) is considered. Hence the algorithm is mapped on hypercube of dimension three. As seen from the pseudocode the iteration is performed for six times for a 8×8 matrix. For every increasing iteration the number of processors used for computation goes on reducing. For the second iteration processors 1 to 6 are used to evaluate for d. For the third iteration 1 to 5 processors are used and the remaining processors in the node are used only for the purpose of routing the data. Processor 8 is always used to evaluate v.

IV. SPEED UP DUE TO MAPPING ON HYPERCUBE

The Householder's method has been successfully mapped on systolic array [10][11][12]. Each processor in systolic architecture for the Householders algorithm communicates with only one processor and in hypercube architecture each processor communicates with three other processor. Hence the speed up which is achieved due to use of broadcasting feature of Hypercube is more practical to use the Hypercube architecture for the real time processing

Let t_f be the time required for one floating point operation (FLOP) and T_d be the time required for a data transfer between two connected nodes of hypercube or the systolic array. Figure 2 gives the graph of total time to perform the computation ($T_f + T_d$) vs number of instructions for both Hypercube and Systolic array for a 8×8 matrix.

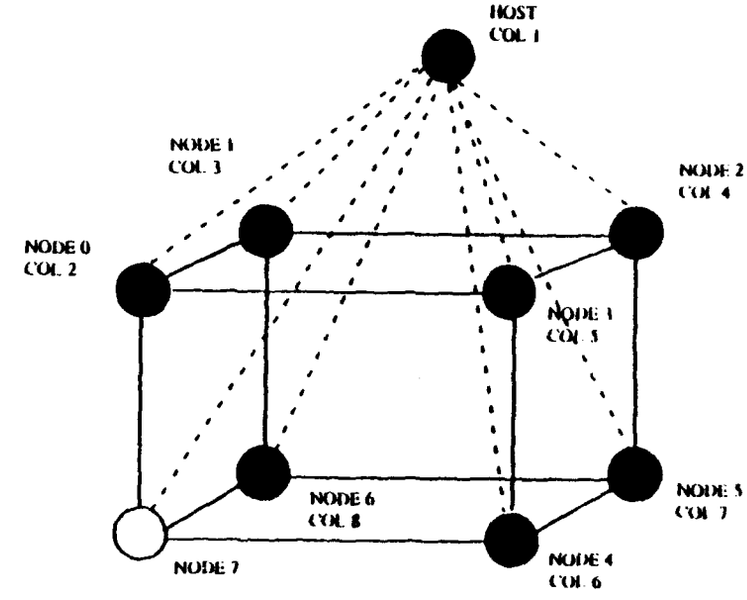


Figure 1: Householders Algorithm for the Matrix of Order 8 Mapped on Hypercube Architecture

V. CONCLUSIONS

As seen from the graph for the matrix of order 8, total time saved is approximately equal to about 70 time units when mapped on hypercube architecture instead of systolic architecture. As the order of the matrix increase the time saved increases. Thus for the matrix of order 2^m, Hypercube mapping is faster than Systolic architecture

REFERENCES

[1] R. O. Schmidt, "Multiple emitter location and signal parameter estimation," IEEE Trans. on Antennas and Propagation, Vol AP-34 No. 3 pp 276-280, Mar. 1986

[2] R. Roy and T. Kailath, "ESPRIT-Estimation of signal parameters via rotational invariance techniques," in proc. IEEE Trans. Acoustic, Speech and Signal Processing, Vol. 37, No. 7, pp. 984-995, July 1989

[3] D. Specht man, A. Paulraj, "Performance analysis of the MUSIC algorithm," in proc. IEEE Conference Acoustic, Speech and Signal Processing, Tokyo, Japan, pp 1909-1912, Apr 1986

[4] T. J. Shan, A. Paulraj, "On smoothed rank profile tests in eigenstructure approach to direction of arrival estimation," in proc. IEEE Conference Acoustic, Speech and Signal Processing, Tokyo, Japan, pp 1905-1908, Apr 1986.
[5] C. Y. Chen and J. A. Abraham, "Fault tolerant systems for computations of eigenvalues and singular value," SPIE Vol 696 Advanced Algorithm and Architecture for Signal Processing, pp 228-237, 1986.
[6] J. J. Dongarra and D. C. Sorensen, "On the implementation of fully parallel algorithm for symmetric eigenvalue problem", SPIE Vol 696 Advanced Algorithm and Architecture for Signal Processing, pp 45-54, 1986.
[7] W. Phillips and W. Robertson, "Systolic architecture for symmetric tridiagonal eigenvalue problem", IEEE International Conference on Systolic Arrays, pp. 145-150, 1988.
[8] J. H. Wilkinson, "The algebraic eigenvalue problem," Clarendon Press, Oxford, Chapter 4, 1965.
[9] M. M. Jawad, S. C. Kwatra, A. H. Djoudi, R. B. Sheelvant, M. S. Rao, "Development of parallel architecture for sensor array processing algorithms" Semi Annual Report submitted to Dept. of Navy, Office of Chief of Naval Research Arlington, VA., Aug 1990.
[10] C. F. T. Tang, K. J. R. Liu and S. A. Tretter, "On systolic array for recursive complex Householder transformations with applications to array processing.", in proc. IEEE Conference Acoustic, Speech and Signal Processing, Toronto, Canada, pp 1033-1036, Apr 1991.
[11] K. J. R. Liu, S. F. Hsieh, K. Yao, "Two level pipelined implementation of systolic block Householder transformations "in proc. IEEE Conference Acoustic, Speech and Signal Processing, pp 1631-1634, Albuquerque, NM, Apr 1990.
[12] S. Y. Kung," VLSI array processors", Prentice Hall, Englewood Cliffs, NJ, 1987.

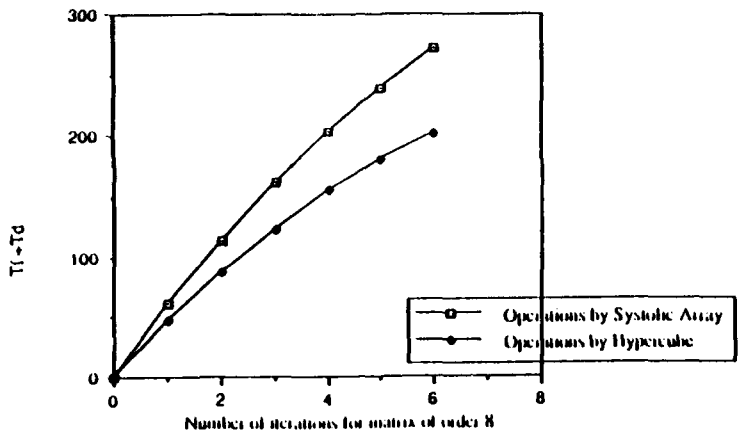


Figure 2. Graph for all the iteration of matrix of order 8.

Modelling of Image Processing Algorithms on Hypercube

D. Kaur[1] A.Yaprak[2] Ahmad. F. Mahmood[3]

[1] University of Toledo, Toledo, OH 43606 [2,3] Western Michigan University Kalamazoo, MI 49008

ABSTRACT

Image Processing Algorithms are very computationally intensive. Therefore, parallel Processing is used in such instances to reduce the processing time. In this paper, hypercube architecture has been utilized to implement image processing algorithms. Image enhancement algorithms viz., edge detection, application of Laplacian operation on image etc., have been implemented on Intel's hypercube simulator ipsc, which can simulate a hypercube of maximum 16 nodes. Various mapping schemes are discussed. The speed up achieved in this process is illustrated.

INTRODUCTION

Of all the parallel architectures proposed to interconnect a large number of processors together, hypercube topology exhibits the best trade off between the number of links required in the network and the diameter of the network[11]. The diameter of a network is the maximum distance needed to reach from a given node to any other node in the network. An *interconnection network with a large diameter has a very low message* passing bandwidth and a network with a high degree of node is very expensive. The product(diameter*degree of a node) is a good criterion to measure the cost and performance of a multiprocessor system. Binary hypercube has a robust topology and has been employed to solve parallel problems[2]. This paper deals with the mapping of image processing algorithms on hypecube. Section I talks about the configuring of hypercube as a 2D mesh to map the picture pixels. Section II discusses the image enhancement algorithms. Section III describes the intel's Personal Supercomputer iPSC.

Image processing algorithms can be divided into three categories: low level, medium level, and high level. Low level vision involves sensing and pre

A Generalized Architecture for DOA Estimation for Wideband/Narrowband Sources

R. Tabar, M.M. Jamali, S.C. Kwatra, A. H. Djouadi

Department of Electrical Engineering
The University of Toledo
Toledo, OH 43606
419-537-2580

ABSTRACT

The high-resolution Direction-Of Arrival (DOA) estimation algorithms are studied to develop architecture for real time applications. Methods for DOA estimation for wideband sources proposed by Buckley and Griffiths[1] and MUSIC [2] algorithm for narrowband sources proposed by Schmidt[2] have been selected for hardware implementation. These algorithms have been simplified and generalized into one common programmable algorithm. It is then parallelized and is executed in a pipelined fashion. A parallel architecture have been designed for this generalized algorithm.

1. INTRODUCTION

Design of special purpose hardware for the implementation of various real time algorithms are possible due to advances in VLSI technology. Pipeline, parallel and distributed processing approaches can be exploited to achieve high throughput rates. In order to develop a real time parallel architecture for a given algorithm following steps need to be performed.

1. An algorithm should be first converted into a computationally efficient algorithm.
2. The selected algorithm is divided into parallel modules.
3. If a particular module of the algorithm can not be executed in parallel due to data dependency it should be pipelined.
4. After parallelizing the algorithm, it should be mapped on a suitable architecture.

This approach of designing special purpose hardware has been applied to the high resolution direction -of -arrival (DOA) estimation for narrowband and wideband cases. The DOA estimation algorithms are based on the processing of the received signal and extracting the desired parameters of the DOA of plane waves. Many approaches have been used for the purpose of implementing the function required for the DOA estimation and are available in the literature[1-6]. The Multiple Signal Classification (MUSIC) is widely studied and provide asymptotically unbiased and efficient estimates of the DOA[1-2]. They estimate the so called signal subspace from the array measurements. The parameters of interest (i.e. determining of the DOA) are then estimated from the intersection between the array manifold and the estimated subspace.
Summary of the MUSIC algorithm is as follows:

1) Estimate the data covariance matrix.
2) Perform the eigendecomposition.
3) Estimate the number of sources.
4) Evaluate Power function.
5) Find the d largest peaks. of Power to obtain estimates of the parameters.

This research is partly supported by ONR grant N00014-91-J-1011
Presented at SPIE's International Symposium on Optical Engineering and Photonics in Aerospace and Remote Sensing April 1993.

First of all MUSIC algorithm has been modified with efficient computational modules. The algorithm has been parallelized. Four modules are implemented using pipeline and parallel processing schemes. First module will compute the covariance matrix. The second and third modules will compute eigenvalues and eigenvectors which will be used by the fourth module. This module computes the power function giving the desired DOA.

For the wideband case, there are many algorithms which are available in the literature. Some of them are extensions of the narrowband cases and others are developed for wideband cases. After reviewing the current literature a wideband approach namely Broad-Band Signal-Subspace Spatial-Spectrum (BASS-ALE) Estimation algorithm proposed by Buckley and Griffiths[1] has been selected for hardware implementation. This algorithm was simplified, pipelined and parallelized. The structure of the algorithm is shown in Figure 1. First, the covariance matrix of the collected data has to be estimated. Then the eigenvalues are computed using the Householder and QR methods[7-12]. From the estimated eigenvalues, an estimation of the signal-subspace dimension D can be calculated according to the steps outlined above. Once the dimension of the system is known, the signal and noise-only eigenvectors can be constructed. The power method is used to find the desired locations θ using the location vector-based estimator.

2. GENERALIZED ALGORITHM AND ARCHITECTURE

The goal of this work is to design an architecture which will be suitable both for narrowband and wideband cases. It can be seen from earlier discussion that the narrowband MUSIC algorithm and the wideband algorithm BASS-ALE requires the similar computational modules such as the computation of the covariance matrix, eigenvalue and eigenvector computation using the Householder transformation and QR method and power method. Since the required modules are identical, they can be generalized into one algorithm and generalized hardware can be developed. The generalized hardware will be suitable for both applications.

First the data has to be collected by the sensors to compute the covariance matrix. In this study eight sensors and eight delay elements have been assumed and hardware is designed accordingly. Eight sensors in the narrowband case will result in a 8*8 covariance matrix. Therefore all computations for DOA estimation will require manipulation of 8*8 matrices. If eight PEs are used in the architecture it would be easier to map the algorithm on these eight PEs. Therefore four modules each using eight PEs can be utilized for this purpose. They will operate in parallel and pipeline fashion.

For the case of the wideband, the BASS-ALE algorithm also requires eight delays in the sensor array. Using eight sensors and eight delays will result in a data vector of size 64. Therefor the computation of the covariance matrix will involve 64*64 matrix. Moreover the Householder transformation, QR method and power method will all operate on 64*64 matrices. It is desired to use eight PEs in each module as that will be sufficient

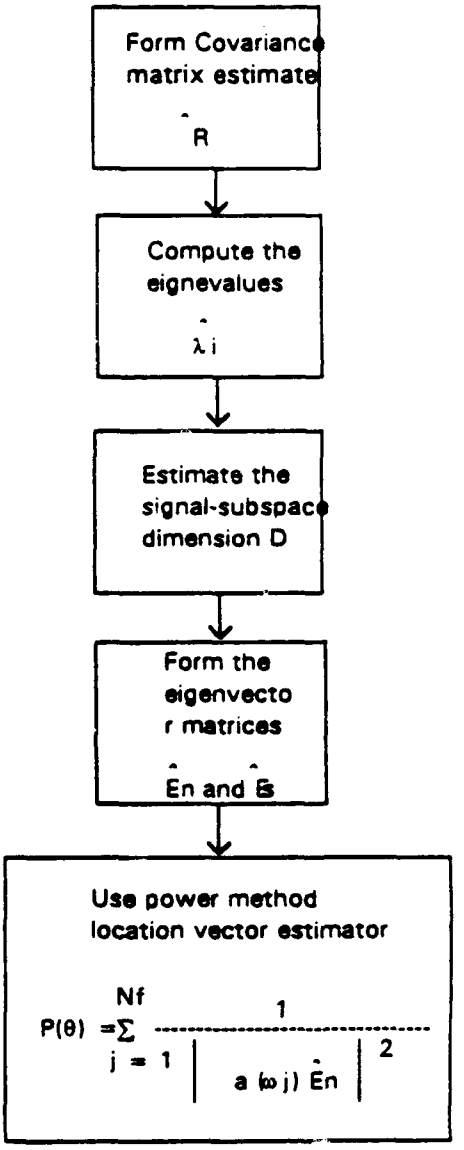


Figure 1 Hardware Units of BASS-ALE Estimation Algorithm.

and will also satisfy the real time requirement. Use of 64 PEs will simply result in an efficient use of the PEs. Use of eight PEs in the manipulation of 64*64 matrices will require proper mapping of the algorithms on these arrays. It is proposed that the covariance matrix computation, Householder transformation , QR method and power method all use modules with eight PEs. In the following subsections, architectures of various modules are described in detail.

3. COMPUTATION OF THE COVARIANCE MATRIX

First of all the data need to be collected by the sensors to compute the covariance matrix . The data output from eight sensors is converted to the digital domain and fed to a pure propagation delay array in a parallel and pipelined fashion as shown in Figure 2. The delay array is implemented using RAM for each sensor output below. The data gathered in the delay array is collected once every eight cycles. In other words, the data is collected every time the array is filled with new vectors. The gathered data is stacked to construct a 64-element data vector required for the computation of the covariance matrix. Computation of the covariance matrix involves a multiplication of 64 element vector with its 64 element complex conjugate transpose (64 element row) producing a 64x64 matrix for each set of data. Since the covariance matrix is symmetric, one way to reduce the required number of computations is to compute only the lower triangular matrix.

A new approach[12] of computing 64x64 covariance matrix using eight processing elements is presented. The elements of the delay array are stacked to create the 64-element data vector. To get more insight about the multiplication process of that vector with its conjugate transpose, a different representation of the data is adopted. Eight sub vectors, eight elements each, are stacked together to form a 64-element vector. The computation of the covariance matrix requires that this 64-element data vector (a column) be multiplied with its counterpart, the 64-element conjugate transpose data vector. The multiplication process creates a Hermitian matrix. On account of creating a lower triangular matrix, 36 sub vector multiplications are required. Each of these sub vector multiplication produces an 8x8 sub matrix. To simplify the computation of the covariance matrix, all the sub vectors will be computed in full including those that are on the diagonal. As a result, more data is generated than is desired, especially the ones above the diagonal. The addition of those extra elements to the matrix establishes a uniform algorithm where all the sub vectors can be multiplied in exactly the same manner without exceptions. In other words, one architecture can be used to compute the 8x8 sub matrices one at a time.

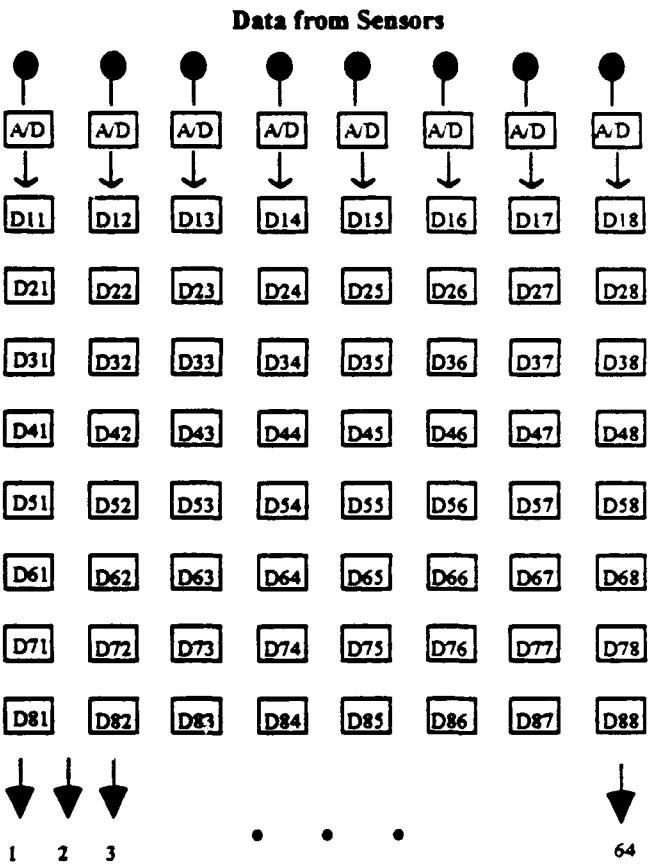


Figure 2 Input data modules

An array of eight PEs are needed to perform the task. The hardware unit computes one of the sub matrices at a time. The broadcast data is stored in the registers and the second operand vector is stored in the PEs. An algorithm to compute the needed 36 sub vector multiplications is provided in the flowchart illustrated in Figure 3. Three counters are needed:

Counter J : Indexes the columns (A column refers to the different sub vectors).
Counter I : Indexes the rows (A row refers to the different sub vectors).
Counter K : Indexes the rows within a sub matrix multiplication process.

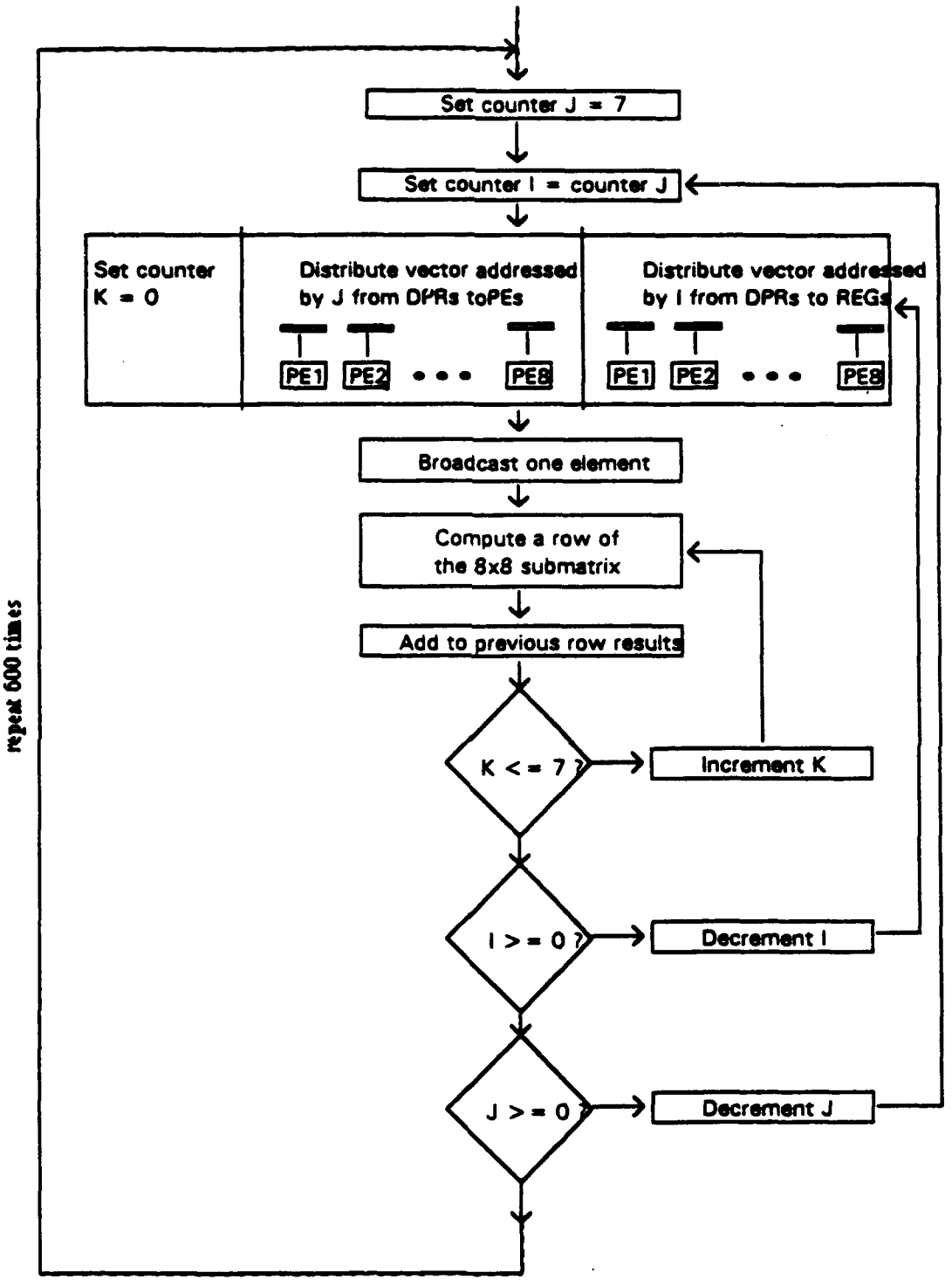


Figure 3 Flowchart illustrating the use of 3 counters to multiply 36 sub matrices forming a lower triangular matrix.

The process proceeds as follows:

Set counter J = 7
Set Counter I = J = 7
All PEs compute in parallel
(a) One operand vector is stored in PEs and is specified by counter J
(b) Second operand vector is broadcasted (specified by K) from a register file and is specified by I.
(c) Perform multiplication of one row in parallel
(d) Products are added to previously stored values.
Decrement the counter I until it reaches 0
Decrement the counter J and set I = J until counter J reaches 0
Repeat for 600 iterations.

Figure 4 shows the needed architecture to perform the operations explained above. Data output from eight sensors are fed and written into eight dual-port RAMs. At any one time, one level of DPRs will be in write mode storing newly read data from the sensors' output while the second level of DPRs will be in read mode where previously stored information is now being used to compute one vector product with its conjugate transpose. Addresses needed by the DPRs are provided by counter I, counter J and a 3-bit counter which is controlled by a 9-bit counter. Two multiplexors, controlled by S (or $\bar{S}$), direct the needed address to the DPRs. If the DPRs are in write mode, the 3-bit address is selected, otherwise, address I and address J are selected. In read mode, the data addressed by counter I is supplied from each DPR to the respective register, while simultaneously the data addressed by counter J is supplied from each of the same DPRs to the respective processing element. Counter K is initialized to the value of zero and then is used to broadcast the output of one of the registers, one at a time, to all of the eight processing elements. Each PE then multiplies that register output data with the value already stored in its internal register (an element of vector J). The multiplication result is added to previously computed values that are stored in memory at an address pointed to by counters I, J and K. Counter K loops through its range (0→7) to construct an 8x8 sub matrix. Counters I and J loop through their range (7→ 0) to compute the 36 sub matrices. At the end of three loops (I, J, K), the assignment of the two levels of DPRs are switched and the operations performed by the PEs are repeated for the newly available data. The process is repeated 600 times to build the required matrix. The covariance matrix is formed by collecting the matrix and dividing its elements by 600.

This architecture can also be used for the computation of the covariance matrix by initializing the counters accordingly. The computations will be very simple and fast as 8 PEs are used for an 8*8 matrix.

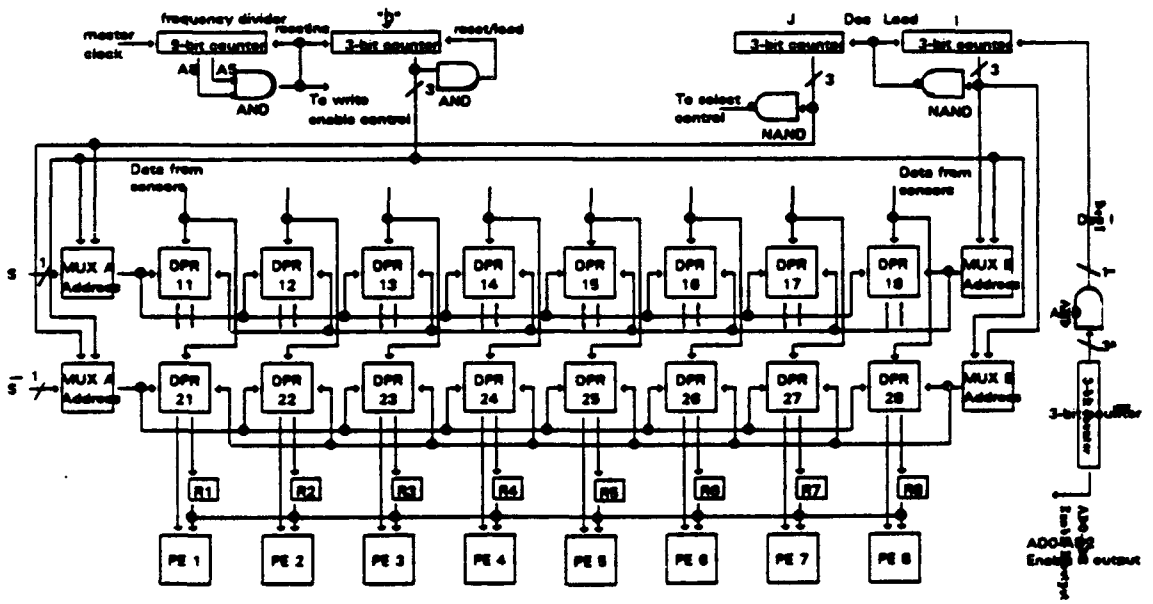


Figure 4 Covariance Matrix Architecture

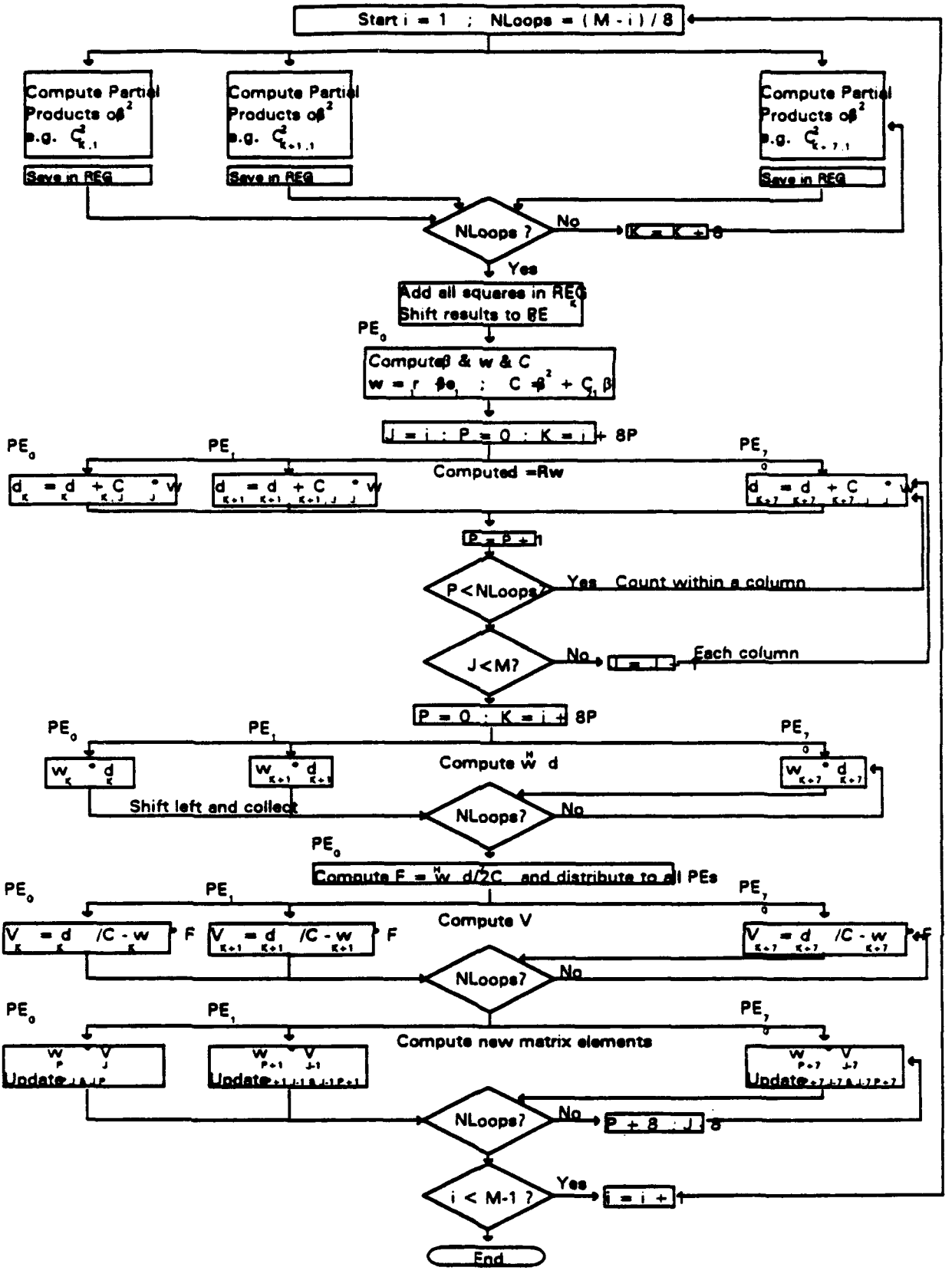


Figure 5 Flowchart for Parallel Householder Transformation

4. HOUSEHOLDER HARDWARE UNIT

The Householder algorithm is chosen to convert the dense covariance matrix, already computed by the previous unit, to a tridiagonal matrix so as to speed up the computations of the eigenvalues/eigenvectors problem. An SIMD architecture is proposed where eight specially designed processing elements are used.

Previous work on the Householder algorithm [12] led to the following simple scheme. Figure 5 shows a flowchart of the steps needed to compute the tridiagonal matrix and it is summarized in the following:

- Compute the scalar value β
- Compute the one dimensional vector **w**
- Compute the scalar c
- Compute the one dimensional vector **d**
- Compute the one dimensional vector **v**
- Modify the Covariance matrix using the above computations

The flowchart portrays the fact that the architecture is not dependent on the dimension (order) of the matrix. The variable NLoops (number of loops) is used to determine the number of loops the system has to undergo repeating the same operations should the order of the matrix be larger than the number of PEs. If NLoops is greater than one, the system will store partial results in the register file of each PE. For example in the narrowband case, the NLoops variable will be set to 1. However in the wideband case, the same variable will be set to a maximum of 8.

Figure 6 shows the proposed SIMD structure for the Householder hardware unit. There are eight processing elements connected through an alignment network to eight blocks of memory (M). Moreover, the PEs have the capability of intercommunications with one another. The type of communication used in this design is a simple link connecting one PE and its neighboring PE. There is one central control unit (CU) with a CU memory core. The alignment network provides the following unique relationship between the PEs and the memory blocks:

- PE_i «—» M_k
- PE_{i+1} «—» M_{k+1}
- M_0 —» all PEs

Note that any processing element (PE_i) can be connected to any memory block (M_k). Subsequently, the linkage of PE_{i+1} is imposed upon M_{k+1} ; hence, the system can have eight different configuration setups. Another possible configuration is the broadcast mode where the memory block M_0 can be connected to all the processing elements to allow sharing of the same information.

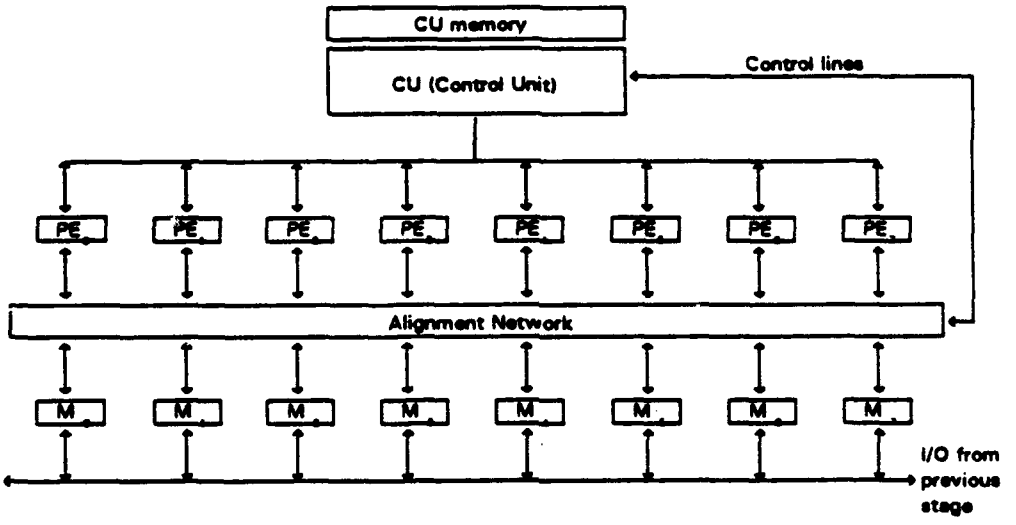


Figure 6 SIMD Architecture for the Householder Unit

For a matrix of order 64, each block of memory should have the capacity to store a little more than $\frac{1}{2}$ K (600) of system words. Each of the covariance matrix's columns, is spread across the memory blocks. When the matrix's order is greater than the number of memory blocks, the remainder of the column is allocated across the memories in a similar pattern. For example, each column of a covariance matrix of order 64 is sectioned into eight subvectors. Each subvector consists of eight elements. Element 0 ($r_{0,j}$) through element 7 ($r_{7,j}$) of the first subvector are stored in memory blocks M_0 through M_7. Element 8 ($r_{8,j}$) through element 15 ($r_{15,j}$) are stored in the succeeding memory cells of the memory blocks M_0 through M_7. The rest of the elements of the column are stored in the memory blocks in the same fashion. Succeeding each column, a row of zeros is allotted for reasons that will become apparent in the following.

The scalar value β can be computed by taking the square root of the length of the vector r_1:

$$\beta^2 = \|r_1\|^2 = r_1^H r_1$$

where

$$r_1 = r_{21}^2 + r_{31}^2 + \ldots + r_{n1}^2 \qquad \text{(where n is the order of the matrix)}$$

The processing elements PE_0 - PE_7 start by reading the first eight values of the vector r_1. The PEs square those elements while reading the next cluster of data. The results of the squared elements are stored concurrently in the register file of each PE. These operations proceed until all of the elements of the r_1 vector are squared and stored in the register file. It is possible that the number of elements in the r_1 vector are not divisible by the number of processing elements, hence, during the last cycle of squaring those elements, some of the PEs will be reading invalid data. To compensate for this kind of difficulty, a row of zeros is inserted between every column of the covariance matrix. Those PEs will be squaring the value zero and, therefore, will have no effect on the calculations. At this instant, the PEs start adding contents of the register file with those of the ALU register. The following illustrates the operations performed by the PEs on a column of 63 entries. Eight steps of reading, squaring and storing data in the register file are required by each PE. At the ninth step, the PEs start adding the previously computed square values stored in the register file to current content of the accumulator (ALU result) register. When all of the registers have been added, the ALU register of each PE holds partial results needed to compute β^2. Partial products are scattered in eight PEs. In order to add them, a shift and add scheme has been proposed. Each PE transmits this information to its neighboring PE on the left through the communication register. *An addition operation is* performed, (PE_0 , PE_2 ,PE_4 , PE_6) hold valid data while the data held by (PE_1 , PE_3 ,PE_5 , PE_7) are neglected. The new results are shifted left twice and another addition operation is performed. Now, PE_0 and PE_4 hold valid data. The last step of the procedure is to shift those results four times to the left and add. The desired value β^2 is now in PE_0 . A simple square root operation is performed to obtain the value β.

After the computation of β, determining the vector **w** is almost trivial. The formula governing the vector **w** is:

$$\mathbf{w} = r_1 + \beta e_1$$

Since all that needs to be done is add the value β to the first element of the vector r_1, a copy of the r_1 vector is transferred to the allocated space in the memory blocks. The vector **w** is ready to be utilized upon addition of β to vector r_1.

The computation of the scalar value c does not cause any complications with the usage of the memory blocks since it does not have to be added to any memory cell. The value c is however, broadcasted to all the PEs via the dump area of the memory blocks; a two step procedure.

The vector **d** is the result of premultiplying the vector **w** with the covariance matrix **R**,

$$\mathbf{d} = \mathbf{R}\mathbf{w}$$

The vector **w** of order 63 is premultiplied with the 63th-order minor of the matrix **R**. The result of the multiplication is a vector **d** of order 63. Every element d_i in the new vector **d** is the scalar product of the ith row of **R** and the vector **w**. In other words, there are 63 multiplications and 62 additions involved to compose each element d_i. Since there are only eight PEs in the architecture, it wouldn't be possible to complete the computations involved for each d_i element in one cycle. In fact, each d_i element need the following calculations,

- Each PE performs eight $r_{i,j}.w_j$ multiplications and stores the results in the register file.
- Each PE adds the results stored in the register file.
- The PEs use the shift left and add scheme discussed earlier to add all the partial results obtained by the individual PEs.

A total of twenty two cycles are required to compute one d_i element. Then, there is the problem of storing one element from one PE into one memory cell in the memory blocks while ignoring the other PEs.

Another approach to this problem would be to compute eight of the **d** vector elements at the same time. It is possible to obtain partial results by driving the PEs to read a partial column J from the minor of the matrix **R**. One element w_j is then distributed to all the PEs and a multiplication operation is performed. At this instant, eight partial results are obtained that can be immediately added to the respective memory locations of the vector **d**. The procedure entails a variable that determines the iteration level of the Householder procedure. Another variable (NLoops) is used to determine the number of loops. If the dimension of the rows of the minor of the matrix **R** is greater than eight, then NLoops is bigger than one and the operations have to be repeated NLoops times. The variable (i) is needed to count those repetitions NLoops times. The value w_j is broadcasted to all the PEs and elements $r_{k,j}$ through $r_{k+7,j}$ are read by the PEs. Each PE multiplies its r.w values and adds the results to the respective memory location of an element d_i. These operations are repeated until all the **d** vector has been calculated. In the narrowband case, where eight is the order of the matrix **R**, there are only eight multiplication and seven addition operations involved per element d_i. Therefore, the available PEs can finish the task in exactly fifteen cycles.

w and **d** are the two constituent vectors of the vector **v** and the formula governing that equation is,

$$\mathbf{v} = \frac{\mathbf{d}}{c} - \mathbf{w}\left(\frac{\mathbf{w}^H\mathbf{d}}{2c^2}\right)$$

The inner product $\frac{\mathbf{w}^H\mathbf{d}}{2c^2}$ (= f) yields a scalar value and can be readily calculated by all the PEs. The partial products in all PEs are shifted and added. The final value f is stored in PE_0. This value f is broadcasted to all the PEs through the dump area in the memory blocks. Assuming that the scalar value c has already been broadcasted to all the PEs, the elements v_i are formed by the following simple formula,

$$v_i = \frac{d_i}{c} - w_i * f$$

The last phase of the householder cycle is to use the new vectors **w** and **v** to update the elements of the matrix **R**. Elements of the matrix **R** that are below or above the subdiagonals are altered to zero and as a result, the matrix **R** transforms into a tridiagonal matrix. The equation for the transformation is,

$$\mathbf{R}_2 = \mathbf{R}_1 - \mathbf{w}\mathbf{v}^H - \mathbf{v}\mathbf{w}^H$$

The vector multiplication $\mathbf{w}\mathbf{v}^H$ and $\mathbf{v}\mathbf{w}^H$ each produce a square matrix of order N-1 or less depending on the iteration level (N is the order of the covariance matrix **R**). As complex matrices, $\mathbf{w}\mathbf{v}^H$ and $\mathbf{v}\mathbf{w}^H$ are the conjugate transposes of each other, where

$$(\mathbf{w}\mathbf{v}^H)^H = \mathbf{v}\mathbf{w}^H$$

The computations of the these complex matrices can be cut in half if the above relationship is taken into consideration. Each PE performs one multiplication w_iv_j and updates two memory locations of the space reserved for the matrix **R**. When the product w_iv_i represents diagonal elements, the PE updates the same memory location twice with the same value. Sometimes, it is possible that the two elements that need to be updated belong to the same memory block, e.g. $r_{1,9}$ and $r_{9,1}$. This does not constitute a problem because the update will take place in two cycles. In other words after the completion of the multiplication w_iv_j, one memory location addressed by row i and column j is updated, an interchange of the addresses i and j follows and the other memory location is updated.

In the same manner that each PE has a different row address index in the first cycle, that PE should also have a different row address index in the second cycle. To achieve this requirement, the plan is to alter the i index in an ascending order (i:=1; i+8; i<=N-1) while altering the j index in a descending order (j:=i+7; j--; j>=i). This scheme is guaranteed not to cause a conflict because no two PEs share a common row or column index.

5. QR HARDWARE UNIT

The QR Hardware Unit is needed to transform the tridiagonal matrix obtained from the Householder Unit into a diagonal matrix. Since this is an iterative method, the resulting matrix is only an estimate of the eigenvalues. Through simulation results and quantization, it is determined that eleven iterations would give reasonable results. Normally, the QR method would take $O(m^3)$ operations, but for a matrix that is already in the tridiagonal form it would require O(m)

operations. If the number of iterations n is more than one, then the number of operations becomes of the order of O(mxn), where m is the order of the matrix and n is the number of iterations. The algorithm is based on Given's Rotations to compute the eigenvalues and the eigenvectors and is illustrated by the following.

$T_1 = T$ {T tridiagonal matrix obtained from Householder }
$U_1 = N^H$ {U represents the eigenvectors matrix }
For i := 1 to n Do
Begin
$R_i = Q_i^H T_i$
$T_{i+1} = R_i Q_i$
$U_{i+1} = Q_i^H U_i$
end; {end For loop}
$\Sigma = T_{n+1}$
$X = U_{n+1}$

After n iterations, the tridiagonal matrix T undergoes a series of transformations and it approximates a diagonal matrix _ whose diagonal elements approach the real eigenvalues of the system. The rows of the matrix X also approach the eigenvectors. Originally, the QR algorithm is a sequential algorithm. Robertson and Phillips[7-11] as well as many others investigated the possibility of modifying the procedure to make it suitable for a parallel environment.

An SIMD architecture with eleven PEs and six memory blocks has been proposed for QR method and is shown in Figure 7 . One PE is dedicated to one iteration, since eleven iterations are assumed, therefore eleven iterations will be executed . The architecture is independent of the size of the matrix and hence can be used both for narrowband and wideband cases. The architecture uses an interconnection network which is able to completely connect six PEs to six memory blocks at any one time.

A closer look at the interconnection network exhibits the following characteristics,

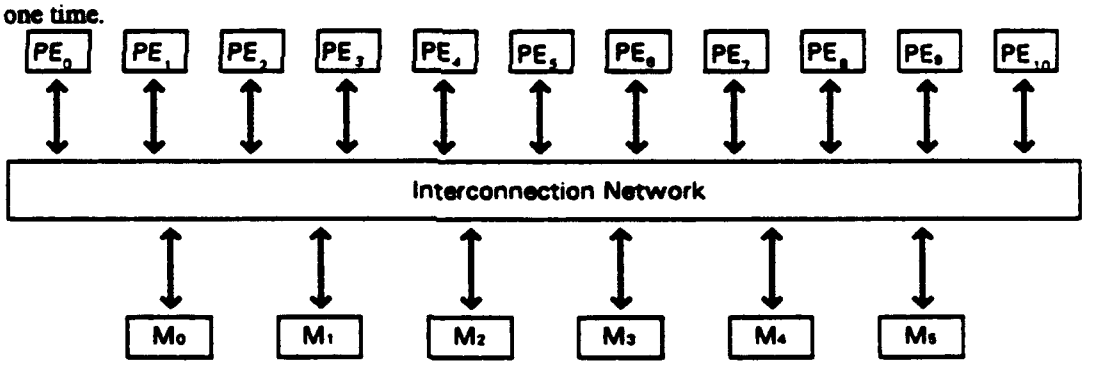


Figure 7 SIMD Architecture for the QR Unit with 32 PEs and 6 Memory Blocks

- Six 2-to-1 Multiplexors needed to switch between processing elements (PE_0, PE_1, PE_2, PE_3, PE_4, PE_5) and (PE_6, PE_7, PE_8, PE_9, PE_{10}) .
- Six 3-to-8 Multiplexors to connect the selected PEs to Memory Blocks M_0 through M_5.
- 1-bit to control the first six MUXs.
- 18-bit control lines to connect the PEs to Ms.

It should also be stressed that at any one time, any six PEs can be completely connected, however, the procedure followed will generate addresses that would require each PE to be connected to a different memory block. In other words, the program is responsible to guarantee a non-blocking situation. One way to remedy the data conflict problem is to write two programs with guaranteed non-conflict cycles. This can be easily accomplished by introducing a delay where some processors will be lagging the others by exactly one cycle. A No-Operation (NOP) instruction will do the trick.

Two PEs share the same pathways to the memory blocks through the interconnection network. One-bit control line is needed to switch between processor PE_i and PE_{i+1}. All the buses are assumed to be a byte wide. Should wider buses be

needed, the appropriate size multiplexors must be used. The eight-bit wide output of the 2-to-1 MUX is directed to the 3-to-8 MUXs. Three-bit control lines are shared by these MUXs to connect one PE to one Bus. Note that the output O_i of each 3-to-8 MUX is tied to BUS_i. The difference of this kind of an interconnection network and a completely connected network is the fact that the former requires 19 control bits while the latter requires 33 control bits.

6. CONCLUSION

A generalized architectures has been developed for the computation of DOA estimation both for narrowband and wideband sources. A MUSIC algorithm was used for the narrowband and BASS-ALE algorithm was selected for the wideband case. First of all an array of 8 Processing Elements (PE) has been used for the computation of covariance matrix and is programmable. Parallel algorithms and parallel architectures for the computation of the eigenvalues and the eigenvectors have been developed and they are also suitable both for narrowband and wideband sources.

7. REFERENCES

1. K. M. Buckley, L.J. Griffiths, "Broad-band signal-subspace spatial-spectrum (BASS-ALE) estimation", IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. 36, July 1988.

2. R.O. Schmith, "Multiple emitter location and signal parameter estimation," IEEE Trans. on Antennas and Propagation, Vol AP-34, No.3, PP. 276-280, Mar. 1986.

3. R. Roy and T. Kailath, "ESPRIT-Estimation of signal parameters via rotational invariance techniques," IEEE Trans. Acoustic, Speech and Signal Processing, Vol. 37, No. 7, PP 984-995, July 1989.

4. D. Spielman, A. Paulraj, "Performance analysis of the MUSIC algorithm," in ICASSP 1986., PP. 1909-1912.

5.T. J. Shan and A. Paulraj, "On smoothed rank profile tests in eigenstructure approach to direction-of-arrival estimation," ICASSP 1986, PP 1905-1908.

6.C.Y.Chen and J. A. Abraham, "Fault-tolerant systems for computations of eigenvalues and singular value" SPIE Vol. 696, Advance Algorithm and architecture for signal processing, pp 228-237, 1986.

7. C.F.T. Tang, K.J.R. Liu, S.A. Tretter, "On systolic array for recursive complex Householder transformations with applications to array processing," ICASSP 1991, PP 1033-1036.

8. S.Y. Kung, "VLSI Signal Processors," Prentice Hall 1987.

9. K.J.R. Liu, S.F. Hsieh, K. Yao, "Two level pipelined implementation of systolic block Householder transformation," ICASSP 1990, PP. 1631-1634.

10. W. Phillips and W. Robertson, "Systolic architecture for symmetric tridiagonal eigenvalue problem," IEEE International Conference on systolic arrays, PP. 145-150, 1988.

11. K.J.R. Liu, K. Yao, "Multiphase systolic architecture for spectral decomposition," 1990 International conference on parallel processing, PP I-123-I-126.

12. M. M. Jamali, S.C. Kwatra, "Development of parallel architectures for sensor array processing algorithms. "Report No. DSPH-2, University of Toledo, 1992.

NAECON 93

MAY 24-28, 1993

93CH3306-8

IEEE

VOLUME 1 OF 2

Design of Special Purpose Parallel Hardware for Real Time Applications

M.M. Jamali, S.C. Kwatra
Department of Electrical Engineering
The University of Toledo
Toledo, OH 43606
419-537-2580

Abstract—A methodology for developing special purpose hardware for real time signal processing applications has been described. The developed methodology is used to explain previously developed architectures for two different applications. First applicaion is for the DOA estimation of sonar signals using the MUSIC algorithm. The second application is for separating large number of channels for on-board satellite communication applications.

I. INTRODUCTION

The growing capabilities in developing very large scale integrated circuits have made possible to design special purpose hardware for the implementation of various algorithms for real time applications [1]. The customized hardware has many advantages which have generated interest among engineers to design application specific special purpose hardware. The design of special purpose hardware will need to exploit pipeline, parallel and distributed processing approaches to achieve high throughput rates. There are many commercially available high speed microprocessors and digital signal processors which can be utilized in a multiprocessor environment. Another approach will be a dedicated circuits may be designed using off the shelf Field Prgrammable Gate Arrays (FPGA). A third approach will be to design a full custom circuit. An appropriate design scheme should be selected for special purpose hardware. The selected approach should be able to exploit maximum parallelization to reduce the computation time.

This research is partly supported by ONR grant number N00014-91-J-1011 and NASA grant number NAG3-799.

II. DESIGN METHODOLOGY

In order to develop a real time parallel architecture for a given algorithm following steps need to be performed.

1. Review the current literature and select an algorithm which requires small amount of computation.

2. The algorithm should be simplified into simple arithmetic operations.

3. The algorithm should be converted into a computationally efficient algorithm or substituted with computationally efficient modules.

4. An estimate should be obtained about the available time for completion of the task.

5. The available time should be used as guide line to select what kind of hardware should be designed and how much parallelism needs to be introduced.

6. An application which requires more multiplications and is compute intensive, an obvious choice should be that a digital signal processor may be selected as they have on chip multipliers and on chip memories. The traditional commercially available microprocessors do not have on chip multipliers.

CH3306-8/93/0000-0054 $1.00 • 1993 IEEE

7. Execution time of the algorithm should be estimated based on the selected processor and available processing speed. If the algorithm can not be executed within the specified time dictated by the application then parallel processing should be considered.

8. The algorithm should be divided into parallel modules. If a particular module of the algorithm can not be executed in parallel due to data dependency it should be pipelined. Degree of parallelizing will also depend on the throughput rate of the system and available time for the computation.

9. After parallelizing the algorithm, it should be mapped on a suitable architecture. The architecture can utilize multiple processors, multiple memories and I/O units. An appropriate architecture should be conceived at this stage. The following guidelines can be used in conceiving an architecture.

(a) If the algorithm consists of repeated simple arithmetic operations, requires very simple programming capabilities and does not have any decision requirements then the architecture should be full custom dedicated architecture. The architecture would consists of group of arithmetic units, memories and required control circuitry.

(b) If the algorithm requires many different operations, elaborate programming, needs decision and conditional statements then the architecture should have a processing element with some memory and other control circuitry.

10. The architecture should then be simulated using a high level language to check the validity of the algorithm and to study the finite precision effects. This exercise will ensure that the algorithm is right and it will also help to identify the required word length.

11. If the algorithm requires multiple processors operating in parallel, then a study should be performed about partitioning of the computing task, synchronization of the data and the mechanism for communication with different processors. It would be wise to consider the amount of data need to transferred from one processor to another. An attempt should be directed that the amount data which need to be transferred should be optimized.

12. The complete structure should be simulated using
Very High Speed Integrated Circuit Hardware Development Language (VHDL) to verify the operation of the all the computational module. It will also be helpful to duplicate the results which were earlier obtained using the high level language.

13. If multiple processors are being used then they should be laid out. Their off chip memory requirement should be minimized. Efforts should be directed to use on chip memory. This approach will help to reduce the execution and communication time. This will also reduce the synchronization requirements.

Design of Special Purpose Hardware for DOA Estimation for Sonar Applications

The high resolution direction-of-arrival (DOA) estimation is important in many sensor systems. It is based on the processing of the received signal and extracting the desired parameters of the DOA of plane waves. Many approaches have been used for the purpose of implementing the function required for the DOA estimation and are available in the literature. The Multiple Signal Classification (MUSIC) and the Estimation of Signal Parameters by Rotational Invariance techniques (ESPRIT) algorithms are widely studied and provide asymptotically unbiased and efficient estimates of the DOA [2-3]. They estimate the so called signal subspace from the array measurements. The parameters of interest (i.e. determining of the DOA) are then estimated from the intersection between the array manifold and the estimated subspace.

There are many algorithms available in the literature which are variations of the MUSIC and ESPRIT algorithms. These

algorithms are derived by varying certain parameters or modifying others at the expense of high computational cost yielding better accuracy etc. In this study MUSIC algorithm has been selected to develop special purpose hardware for real time applications. Although MUSIC is a high resolution algorithm, it has several drawbacks including the fact that complete knowledge of the array manifold is required, and that is computationaly very expensive as it requires a lot of computations to find the intersection between the array manifold and the signal subspace. This drawback can be overcome by using high speed parallel architecture for the computation of DOA.

It was decided that special purpose hardware can be developed which can be used for real time computation of the DOA estimation. First of all computation requirement of this algorithm were investigated [4]. Following is list of operations which need to be performed.

1) Estimate the data covariance matrix.
2) Perform the eigendecomposition.
3) Estimate the number of sources.
4) Evaluate Power function.
5) Find the largest peaks of Power and estimate direction of arrival.

In order to develop a real time parallel architecture for a given algorithm following steps need to be performed.

a) Estimate the computation requirement of the algorithm. This is dependent on the number of sensors used in the algorithm. After reviewing the literature it was decided that eight sensors should be sufficient for this application. This was also discussed with one of the experts in the area at one of the conference. It was confirmed that use of eight sensors was a reasonable assumption.

b) The second task was what would be considered as the real time for this application. Again a search was conducted in the literature about the frequency range of the sonar signals. It was found that frequency of 27 KHz is a reasonable number for this application.

c) The third task was how much data need to be collected which will give reasonable accurate results. It was found that collection of 4800 samples will give reasonable results.

d) The number of samples to be collected (4800) and the signal coming at upto 27 KHz. will give the available time for completing the task and giving the available time. A sampling frequency which should be twice the signal need to be used. Therefore a sampling frequency of 100 KHz and 4800 samples used as a guideline which will give allowed computation time of 48 m secs.

e) An algorithm should be first converted into a computationally efficient algorithm. It is clear from the above discussion that the implementation of the MUSIC algorithms requires formation of data covariance matrix and the computation of the eigenvalues. It was found that QR algorithm is very appropriate for the computation of the eigenvalues and the eigenvectors. To make this algorithm more efficient it was decided to use the Householder method to convert the covariance matrix into tridiagonal one before performing QR decomposition. This approach of reducing the symmetrical covariance matrix into tridiagonal matrix make QR algorithm more efficient. The eigenvalues are used to find the number of sources. Finally using the eigenvectors in Power method the directions of arrival are computed.

Using the above time limit of 48 m secs and eight sources, it was found that four major computation intensive operations need to be performed. These algorithms also require multiplication, division, square root, logarithm, sine & cosine and square operations. It can be seen that the computation requirements are extreme and many complex operations need to performed as opposed to simple addition/ subtraction operation. Therefore use of powerful digital signal processor will be appropriate. Since many operations need to performed in parallel requiring multiple processors, therefore eight processors should be connected in parallel.

Since the algorithm requires computation of four major computational tasks, therefore four computational units each with eight processor can be used. These computational units will operate in parallel. The data will be passed from one unit to another in a pipeline fashion. Therefore four computational module will execute this algorithm in parallel and pipeline fashion. The MUSIC algorithm has been simulated using the FORTRAN language and appropriate results have been obtained. This was done to validate the modifications which were made in this algorithm [4]. The simulation assumed infinite precision. Efforts are being directed to simulate this architecture using the assembly language of the Motorola DSP 56000 digital signal processor.

Special Purpose Hardware Development for Satellite Applications

Due to the growth and extreme demand in the area of mobile communication, it may be necessary that the future satellite communication systems should provide service to large number of small capacity, multi service users. For these systems the conventional transmission methods of Frequency Division Multiple Access (FDMA) or Time Division Multiple Access (TDMA) are not efficient. One approach to offer these services at a low cost to the user is to use Single Channel Per Carrier (SCPC)/FDMA on the uplink and Time Division Multiplexing (TDM) on the downlink [5-6]. This approach will result in low cost and less complex earth terminals, enabling an increase in communication via satellites. The problem with this type of communication is that it transfers the burden of computation on-board the satellite, where power and area requirements are critical. It can thus be seen that hardware that is efficient in terms of speed, power consumption and components needs to be developed for performing the computations on-board the satellite.

One of the major components in the FDMA/TDM conversion is the transmultiplexer which is required to separate the FDMA signal into individual channels. Using today's technology special purpose full custom VLSI digital circuit can be designed for the transmultiplexer.The Digital transmultiplexer will provide flexibility in processing any number of channels with different bandwidths and access formats. Moreover the transmultiplexer may be programmable and programming instructions may be send from the ground for the desired processing.

There are two goals for the digital implementation of the transmultiplexer. First of all it should consume small amount of power and the second one that it should meet the real time processing requirement of the system. In this work a case study of the design of a programmable TMUX is presented. It was determined that the TMUX should be capable of demultiplexing 800 channels at 64 Kb/s each. The design can later on be extended to accommodate varying number of channels with varying bit rate.

It has been shown earlier that the most efficient type of TMUX for demultiplexing channels of uniform bandwidth is the polyphase FFT method [7]. It performs both sampling rate reduction (decimation) and channel separation together. The decimation is implemented using the commutator and a set of filters. The input samples are fed to the polyphase filters by the commutator. The outputs of the DFT operation are multiplied by another constant (-1)m (where m is the decimated sample number) to obtain the separated channels samples at baseband. In this approach polyphase filters are derived from a prototype filter. The polyphase filter coefficients are obtained from the polyphase decomposition of the low pass prototype filter. In this case the decimation is performed before the filtering, therefore the filtering is performed at the lower rate. The detailed derivation of the polyphase FFT principles is given in [7-9].

The FDMA signal is complex sampled at F MHz. For a sampling rate of F MHz the system clock period will be less than T secs (1/F Hz). Two A/D converters are needed, one for the real part and one for the imaginary part. The samples are fed into the

polyphase network at the rate of one complex sample per channel every T secs.

For real time demultiplexing of 800 channels each having a bandwidth of 45 KHz, the polyphase filtering and the FFT operations should be completed in t = 1/45 K Hz secs or 22.22 micro seconds. This is because the sample interval after decimation is t secs. Performing both these operations (filtering and FFT) in t secs will require very high speed hardware and may not be practical. This problem is solved by performing a data analysis and realizing that the two operations are data independent and can therefore be performed by two modules in a pipelined fashion, thus giving each module t secs to perform its operations. This method of pipelining the two operations allows the data to come in continuously and facilitates the separation of channels in real time. It also avoids buffering and accommodates other necessary operations such as multiplication by a constant factor for phase shifting operations. The two operations are implemented as two modules, namely, the polyphase filter module and the FFT module. The two modules are explained in the following sections.

A bank of polyphase filters is required for performing the weighting operation. Each filter in the polyphase network is derived from the prototype lowpass filter [8-9]. In this case the prototype filter has 7200 taps and there are 800 channels to be demultiplexed, the 800 polyphase filters would have 9 taps each. The polyphase filters are implemented as FIR filters [8-9].

The individual 9 tap FIR filters can be implemented in a variety of ways. For filtering the proposed 800 channels, each 9 tap filter will have 22.22 micro secs to compute its result. This can be achieved with today's technology, but implementing 800 filters and assuming that each filter has only one multiplier, the total power requirements will be very high.

It can be visualized that filtering operation consists of repeated multiplication and addition operations. The filtering operation also require very little programming capabilities. Therefore a full custom dedicated architecture would be more appropriate for this application. The dedicated architecture would give better speed/power performance as compared to commercially available digital signal processors.

A structure of 9 tap filter which will be shared amongst the 800 filters is proposed. This is accomplished by designing the hardware structure for one 9 tap filter and passing the appropriate coefficients that define a desired filter. The coefficients are stored in a high speed memory. Using this approach any desired filter can be configured by accessing the appropriate filter coefficients from memory. Thus in this architecture the structure for one filter is time shared amongst the 800 filters giving an approximately 800:1 savings in hardware and power consumption over directly implementing the polyphase structure.

The architecture of the shared polyphase filter bank for the case of 800 (9 tap) filters is shown in Figure 2. In this structure each high speed RAM corresponds to one of the registers in the 9 tap filter. In essence the RAMs used in the current structure act as multiple registers connected to multipliers. An address generator generates the read and write addresses by counting from 1 to 800. For 800 polyphase filters the counter will count till 800. The read and write addresses are the same through one cycle of reading and writing.

Outputs of the filter bank are passed through a DFT to compensate for the phase shifts due to the polyphase filters. It can be seen that 800 samples will be available for DFT computation every 22.22 micro secs. The FFT algorithm is used for fast computation of the DFT. For efficient FFT implementation, an N point FFT (closest power of 2 above X) is computed instead of an X point FFT. Normally zeroes are added to the data points to get N points. For example, demultiplexing 800 channels with a 45 KHz bandwidth each would require a 1024 point FFT to be completed in 22.22 μsecs (1/45 KHz). As there are no DSPs that can perform the FFT at the high speeds required with reasonable power consumption,

dedicated hardware must be designed to perform the FFT operation efficiently. Moreover FFT algorithm consists of regular structure with repetitive multiplication and addition operations and requires very little programming capabilities, therefore a dedicated architecture consisting arithmetic units specially designed for butterfly operations, memories to store intermediate data would be more appropriate.

One method for implementing the FFT processor is the pipelined FFT [10-11]. In this architecture there will be log2N stages in the pipeline. Each stage in the pipeline will consist of a Arithmetic Element(AE), a memory that will store the twiddle factors, a RAM for storing its output and an address generator. Since the FFT algorithm requires multiplication of data with different twiddle factors at different stages, this would necessitates simultaneous reading and writing of two intermediate results. Therefor it would be appropriate to use two memories which would facilitate simultaneous reading and writing. The RAM that stores the output of each stage is divided into two sections. Section 1 ranges from 0 to 1K and section 2 varies from 1 to 2K. The memory is clearly partitioned into two parts to enable simultaneous reading out of and writing into different sections of the same memory. The nth RAM stores the result of the (n-1)th stage of the FFT.

Each stage of the FFT requires 512 butterflies. Each butterfly requires a pair of inputs and one twiddle factor. All the butterfly operations required in each stage of the FFT pipeline are performed sequentially. Each stage of the FFT has a different set of input data addresses than the other stage. The 512 twiddle factors needed for a 1024 point FFT are stored in an 512 word by 16 bit (8 bits each for the real and imaginary parts) ROM called a Coefficient ROM (CR).

An arithmetic element has been designed to compute the butterfly operations. Each stage will have one arithmetic element.

IV. CONCLUSIONS

A methodology for designing special purpose hardware for real time applications have been presented. The developed methodology has been applied to two different applications. First application is the DOA estimation using the MUSIC algorithm. The MUSIC algorithm has been modified with efficient computational modules. The algorithm has been parallelized. Four modules are implemented using pipeline and parallel processing schemes. The architecture is developed assuming eight sensors and is capable of computing DOA in real time. The architecture utilizes identical processing elements.

The second application discussed was the development of efficient architecture for demultiplexing a varying number of uniformly spaced channels is given. The design is divided into two parts, the polyphase filter bank implementation and the FFT implementation. The polyphase filter bank is implemented by means of a parallel, pipelined and multiplexed shared filter module. In this shared filter bank module the hardware structure for one filter is shared amongst the 800 different polyphase filters giving rise to an 800 to 1 savings over directly implementing the N polyphase filters. For FFT computation, a parallel and pipelined implementation is used.

V. REFERENCES

1. S.Y. Kung,"VLSI Signal Processors," Prentice Hall 1987.

2. R.O. Schmith,"Multiple emitter location and signal parameter estimation," IEEE Trans. on Antennas and Propagation, Vol AP-34, No.3,PP. 276-280, Mar. 1986.

3. R. Roy and T. Kailath,"ESPRIT-Estimation of signal parameters via rotational invariance techniques," IEEE Trans. Acoustic,Speech and Signal Processing, Vol. 37, No. 7, PP 984-995, July 1989.

4. M. M. Jamali, S.C. Kwatra,"Development of parallel architectures for sensor array processing algorithms."Report No. DSPH-2, University of Toledo, 1992.

[5] B.G.Evans and F.P. Coakley, "Multi carrier demodulators for OBP satellites," International Journal of Satellite Communication, pp. 243-251, 1988.

[6] J. Campanella and S. Sayegh, "On-board multichannel demultiplexer/demodulator" NASA Report, 1987.

[7] M. Bellanger and P. Daguet , "TDM/FDM transmultiplexers: digital polyphase and FFT," IEEE Transactions on Communication , pp. 1199-1205, Sep. 1974.

[8] P. J. Fernandes,"A parallel/pipeline architecture for a reconfigurable transmultiplexer,"M.S. Thesis, University of Toledo, Toledo, Ohio, February 1991.

[9] C. Gottschalk,"The ASIC Design of a Digital Transmultiplexer Using Standard Cell Libraries," M.S. Thesis, University of Toledo, Toledo, Ohio, August 1992.

[10] H. S. Stone, "Parallel processing with the perfect shuffle," IEEE Transactions on Computers, pp. 153-161, Feb. 1971.

[11] S. I. Sayegh, "A Pipeline Processor for Mixed-Size FFT's," IEEE Trans. on Signal Processing, PP.1890-1990, August 1992.

An Architecture for DOA Estimation for Broad-Band Sources

R. Tabar, M.M. Jamali, S.C. Kwatra
Department of Electrical Engineering
The University of Toledo
Toledo, OH 43606
419-537-2580

Abstract - The high-resolution Direction-Of-Arrival (DOA) estimation is important in many sensory systems like radars, sonars and seismic exploration. With the advances in the area of VLSI it is now possible to design a special purpose hardware for DOA estimation which will be suitable for real-time sonar applications. One of the methods for DOA estimation for broad-band sources proposed by Buckley and Griffiths [4] has been selected for hardware implementation. This BASS-ALE algorithm has been simplified, parallelized and is mapped on an architecture suitable for real time processing.

I. INTRODUCTION

With the advancement of very large scale integration, more circuitry can be compacted on a single integrated circuit and now more and more Application Specific Integrated Circuits (ASIC) are being developed. One area where ASIC approach can be applied is the Direction of Arrival (DOA) estimation for sonar applications. Special purpose hardware can be designed for complex algorithms exploiting parallel processing approaches in the form of ASICs for real time applications.

The problem of multiple source location is of interest to us and has been investigated in the literature. Many of the common methods of source location are based on narrow-band assumptions on the signals. There are many broad-band DOA estimation algorithms available in the literature[1..6]. Some of them are extensions of narrow-band cases and others are transformed to specific broad-band algorithms. One of the broad-band methods proposed by Buckley and Griffiths [4] uses a focused covariance matrix as a temporal/spatial focused observation for broad-band source representation. BASS-ALE estimators employ the eigenstructure of broad-band data covariance matrix and broad-band models. BASS-ALE algorithm has been selected for the design of special purpose hardware. It has been simplified and parallelized [12]. Eight sensors and eight delays are assumed in this work for the computation of DOA. The algorithm will be executed in a pipeline fashion. First of all the covariance matrix will be computed from the collected data followed by eigenvalue/eigenvector computations. The eigenvalues and eigenvectors will be computed using Householder method followed by QR method. Dimension of the span of the signal-subspace spatial-spectrum will be evaluated. Finally, a power method will be used to compute DOA.

This research is partly supported by ONR grant number N00014-91-J-1011.

Accepted for presentation at Oceans '93

II. ARCHITECTURE FOR BASS-ALE ALGORITHM

First of all the data need to be collected by the sensors to compute the covariance matrix. The data output from eight sensors is converted to the digital domain and fed to a pure propagation delay array in a parallel and pipelined fashion. The delay array is implemented using RAM for each sensor output.

A. The Covariance Matrix Hardware Unit

The data gathered in the delay array is collected once every eight cycles. In other words, the data is collected every time the array is filled with new vectors. The gathered data is stacked to construct a 64-element data vector required for the computation of the covariance matrix. Computation of the covariance matrix involves a multiplication of 64 element vector with its 64 element complex conjugate transpose (64 element row) producing a 64x64 matrix for each set of data. Since the covariance matrix is symmetric, one way to reduce the required number of computations is to compute only the lower triangular matrix.

An approach of computing 64x64 covariance matrix is presented and uses eight processing elements. The computation of the covariance matrix requires that the 64-element data vector (a column) be multiplied with its counterpart, the 64-element conjugate transpose data vector. The multiplication process creates a Hermitian matrix. On account of creating a lower triangular matrix, 36 sub vector multiplications are required. Each of these sub vector multiplication produces an 8x8 sub matrix. To simplify the computation of the covariance matrix, all the sub vectors will be computed in full including those that

are on the diagonal. As a result, more data is generated than is desired, especially the ones above the diagonal.

The addition of those extra elements to the matrix establishes a uniform algorithm where all the sub vectors can be multiplied in exactly the same manner without exceptions. In other words, one architecture can be used to compute the 8x8 sub matrices one at a time. The hardware unit computes one of the sub matrices at a time. The broadcast data is stored in the registers and the second operand vector is stored in the PEs. Three counters are needed:

Counter J : Indexes the columns (A column refers to the different sub vectors)

Counter I : Indexes the rows (A row refers to the different sub vectors)

Counter K : Indexes the rows within a sub matrix multiplication process.

Fig. 1. shows the needed architecture to perform the operations explained above. Data output from eight sensors are fed and written into eight dual-port RAMs (DPR). At any one time, one level of DPRs will be in write mode storing newly read data from the sensors' output while the second level of DPRs will be in read mode where previously stored information is now being used to compute one vector product with its conjugate transpose. Addresses needed by the DPRs are provided by counter I, counter J and a 3-bit counter which is controlled by a 9-bit counter. Two multiplexors controlled by S (or S), direct the needed address to the DPRs. If the DPRs are in write mode, the 3-bit address is selected, otherwise, address I and address J are selected. In read mode, the data addressed by counter I is supplied from each DPR to the respective register, while simultaneously the data addressed by counter J is supplied from each of the same DPRs to the respective processing element. Counter K is initialized to the value of zero and then is used to broadcast the output of one of the registers, one at a time, to all of the eight processing elements. Each PE then multiplies that register output data with the value already stored in its internal register (an element of vector J). The multiplication result is added to previously computed values that are stored in memory at an address pointed to by counters I, J and K. Counter K loops through its range (0→7) to construct an 8x8 sub matrix. Counters I and J loop through their range (7→0) to compute the 36 sub matrices. At the end of three loops (I, J, K), the assignment of the two levels of DPRs are switched and the operations performed by the PEs are repeated for the newly available data The process is repeated 600 times to build the required matrix. The covariance matrix is formed by collecting the matrix and dividing its elements by 600.

B. Householder Hardware Unit

The Householder algorithm is chosen to convert the dense covariance matrix, already computed by the previous unit, to a tridiagonal matrix so as to speed up the computations of the eigenvalues/eigenvectors problem. An SIMD architecture is proposed where eight specially designed processing elements are used.

Previous work on the Householder algorithm [11] led to the following simple scheme:

- Compute the scalar value β
- Compute the one dimensional vector w
- Compute the scalar c
- Compute the one dimensional vector d
- Compute the one dimensional vector v
- Modify the covariance matrix using the above computations

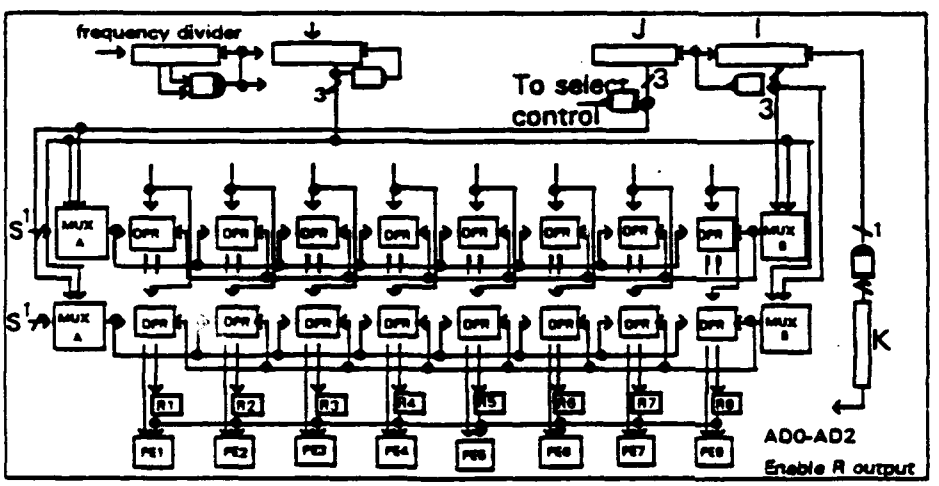


Fig.1. The Covariance matrix architecture.

The architecture is not dependent on the dimension (order) of the matrix. Fig. 2. shows the proposed SIMD structure for the Householder hardware unit. There are eight processing elements connected through an alignment network to eight blocks of memory (M). Moreover, the PEs have the capability of intercommunications with one another. The type of communication used in this design is a simple link connecting one PE and its neighboring PE. There is one central control unit (CU) with a CU memory core. The alignment

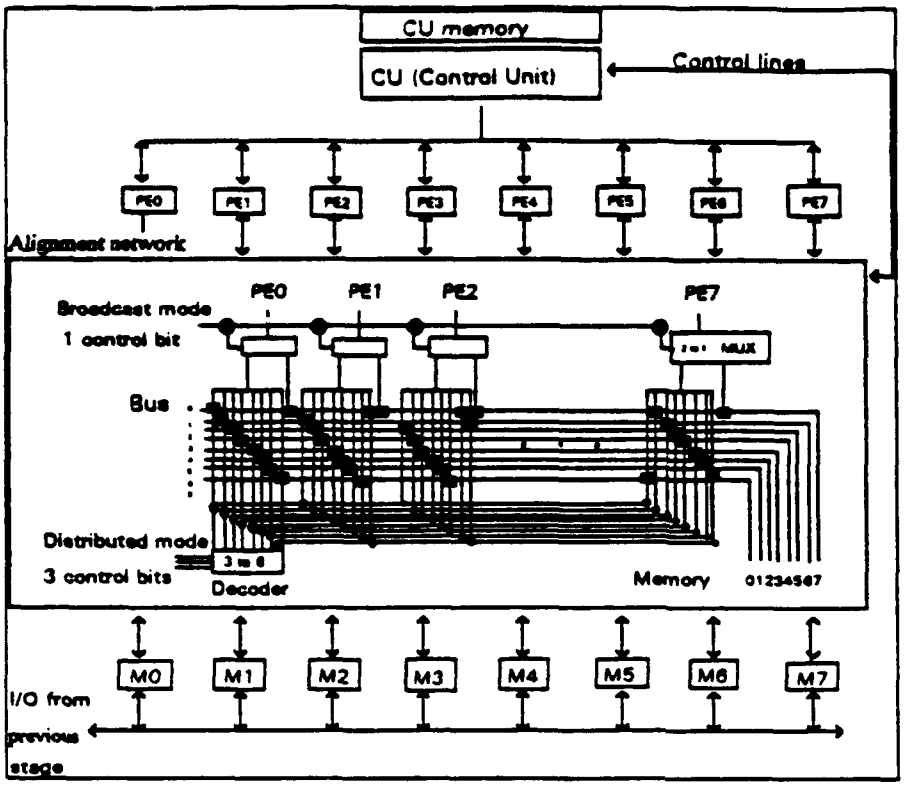


Fig. 2. Householder unit architecture.

network provides the following *unique relationship* between the PEs and the memory blocks:

- PE_i <—> M_k
- PE_{i+1} <—> M_{k+1}
- M_0 —> all PEs

Note that any processing element (PE_i) can be connected to any memory block (M_k). Subsequently, the linkage of PE_{i+1} is imposed upon M_{k+1} ; hence, the system can have eight different configuration setups. Another possible configuration is the broadcast mode where the memory block M_0 can be connected to all the processing elements to allow sharing of the same information. One control-bit is used to put the system in the broadcast mode or in normal-operation mode. This is achieved through the use of a 2-to-1 multiplexor (MUX) per PE to connect to either bus_0 or bus_i. When the system is in normal-operation mode, a 3-to-8 decoder channels the flow of data on the buses to the PEs using transmission gates (TG). These TGs are numbered 1 to 8 for each PE. Note that TG_1 of each PE connects PE_i to bus_i, TG_2 connects PE_i to bus_{i+1}, and so on. This scheme conforms with the requirement that if PE_i is connected to M_k, then it follows that PE_{i+1} should be connected to M_{k+1}.

Each block of memory is sectioned to allocate space for the covariance matrix, a dump area, reserved space, part of the w vector, part of the d vector and part of the v vector. Each sub vector consists of eight elements. Element 0 ($r_{0,j}$) through element 7 ($r_{7,j}$) of the first sub vector are stored in memory blocks M_0 through M_7. Element 8 ($r_{8,j}$) through element 15 ($r_{15,j}$) are stored in the succeeding memory cells of the memory blocks M_0 through M_7. The rest of the elements of the column are stored in the memory blocks in the same fashion.

C. QR Hardware Unit

The QR hardware unit is needed to transform the tridiagonal matrix obtained from the Householder unit into a diagonal matrix. Since this is an iterative method, the resulting matrix is only an estimate of the eigenvalues. Through simulation results and a study of the margin of error it is determined that eleven iterations would give reasonable results. Normally, the QR method would take $O(m^3)$ operations, but for a matrix that is already in the tridiagonal form it would require O(m) operations. If the number of iterations n is more than one, then the number of operations becomes of the order of O(mxn), where m is the order of the matrix and n is the number of iterations. The algorithm is based on Given's rotations to compute the eigenvalues and the eigenvectors and is illustrated by the following,

$T_1 = T_H$ {T tridiagonal matrix from Householder}
$U_1 = N$ {U represents the eigenvectors matrix}
For i := 1 to n Do
Begin

$$R_i = Q_i^H T_i$$

$$T_{i+1} = R_i Q_i$$

$$U_{i+1} = Q_i^H U_i$$

end; {end For loop}
$\Sigma = T_{n+1}$
$X = U_{n+1}$ (1)

After n iterations, the tridiagonal matrix T undergoes a series of transformations and it approximates a diagonal

matrix Σ whose diagonal elements approach the real eigenvalues of the system. The rows of the matrix X also approach the eigenvectors.

Parallel QR algorithm

Originally, the QR algorithm is a sequential algorithm. Robertson and Phillips [9] as well as many others investigated the possibility of modifying the procedure to make it suitable for a parallel environment. The approach described by Robertson and Phillips [9] is adopted in this work.

Let a(j,i) and b(j,i) be the entries of the diagonal and the subdiagonal of the matrix T, respectively, where j is the row/column number and i is the iteration number. Moreover, carefully observing the operations involved in the sequential algorithm, it becomes apparent that, in the same iteration level, each computation of an a(j+1,i) or b(j+1,i) entry depends on its preceding entries. However, the computation of any entry a(j,i+1) or b(j,i+1) depends also on the results obtained from the calculations of the entries a(j,i) and a(j-1,i) or b(j,i) and b(j-1,i).

In this instance, each column comprises of computations of all the elements a(j,i) or (b(j,i) (For j:= 1 to 64) in the same iteration level. Since an element of the succeeding iteration level (j,i+1) depends on the previous two elements from the previous level, (j,i) and (j-1,i), the computations of the entries (j,i+1) for (For j:= 1 to 64) can start at a two steps delay. In this configuration, each PE is assigned the computations of all the entries that fall in the same iteration level. The total number of steps a PE is active depends on the order of the matrix, hence, in the above example, 64 is the number of active steps. Bearing in mind that the number of iterations is eleven, the number of PEs needed is dependent on the number of iterations. The total number of steps required for the completion of the algorithm in the same example of the matrix of order 64 is also 84. It was verified from the simulation that 11 iterations will be needed, therefore in our architecture 11 PEs are suggested.

It can be shown that six memory blocks will suffice to complete the architecture. Fig. 3. shows an architecture using an interconnection network as the exchange data environment.

It should also be stressed that at any one time, any six PEs can be completely connected, however, the procedure followed will generate addresses that would require each PE to be connected to a different memory block. In other words, the program is responsible to guarantee a non-blocking situation. One way to remedy the data conflict problem is to write two programs with guaranteed non-conflict cycles. This can be easily accomplished by introducing a delay where some processors will be lagging the others by exactly one cycle. A No-Operation (NOP) instruction will do the trick.

D. Dimension of the span of the signal-subspace spatial-spectrum

In the previous architectural units, tasks are to collect the data constituted of the sample signal in white noise surroundings, to produce the covariance matrix, and to perform an eigenvalue/eigenvector analysis. The eigenvalues λ_i are sorted in a monotonically decreasing order. It is however desired to detect the dimension of the space that spans the signal space and the dimension that spans the noise space. The Akaike information criterion estimate (AICE) is an approach used to determine the number of signal-subspace parameters d needed in such an estimation problem. It is required to choose the number d

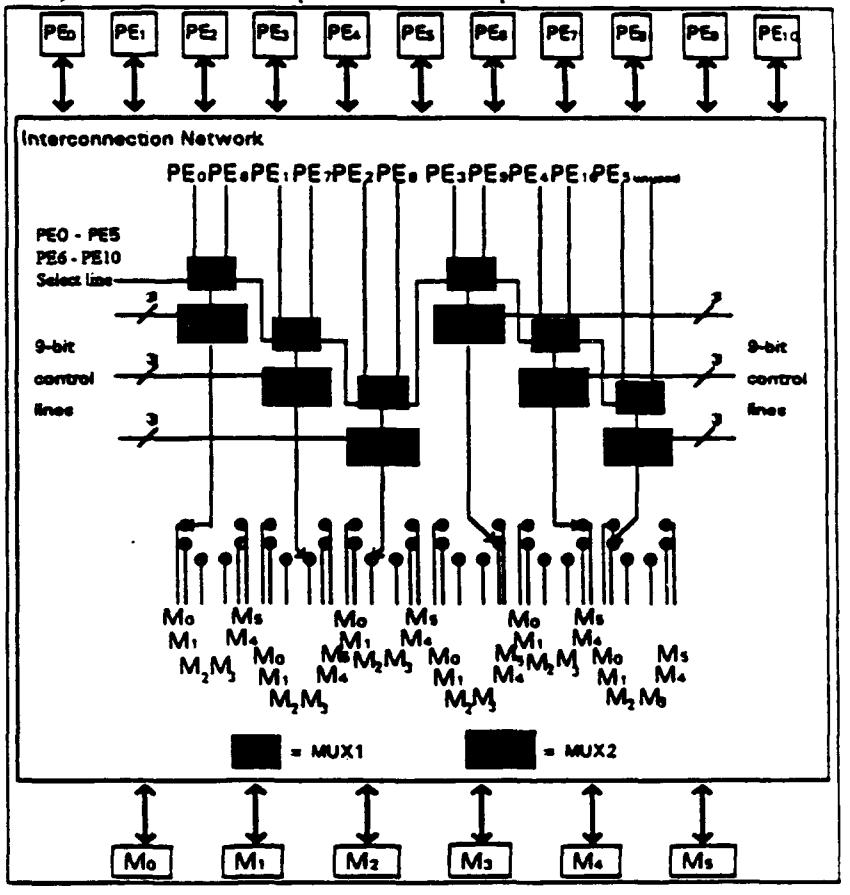


Fig. 3. QR unit architecture.

which minimizes the function AICE(d) defined by [4] is described:

$$AICE(d) = -2 \log \left[\begin{array}{l} \text{maximum of the likelihood function} \\ \text{of the observation obtained} \\ \text{by changing the d free parameters} \\ \text{in the prespecified model} \end{array} \right] + 2d$$

The AICE condition as applied to the BASS-ALE algorithm yields the following Equations,

$$AICE(d) = (J_2 - J_1 + 1).N.(ML - d).LN\left(\frac{a_0}{g_0}\right) + d(2ML - d) \quad (2a)$$

where

$$a_0 = \frac{1}{ML - d} \sum_{i=d+1}^{ML} \lambda_i \quad (2b)$$

$$g_0 = \left(\prod_{i=d+1}^{ML} \lambda_i \right)^{\frac{1}{ML-d}} \quad (2c)$$

J_1, J_2 such that $0 \le J_1 < J_2 \le J-1$ and

J = KL M-dimensional snapshot vectors and K equals the number of contiguous blocks of vectors.

As is, the above equation might create some difficulties in the computations due to summation and multiplication in a_0 and g_0. A new scheme is proposed [12] that takes advantage of the recursive nature of this equation. A count down loop structure will start the summation process so that at every iteration, only one eigenvalue λ will be added to previously summed eigenvalues. This technique eliminates repeating the same computations again and again thus saving on computation time and makes it efficient for real-time processing.

E. The Power Method

After computing the eigenvalues and eigenvectors, the signal-subspace order estimation as described in [4] is accomplished in the SSD module. Using the signal subspace dimension, the noise eigenvectors can be used to compute the following function,

$$P(\theta) = \sum_{j=1}^{N_\xi} \frac{1}{| a(\omega_j) \hat{E}_n |^2} \quad (3)$$

where $\hat{E}_n$ are the eigenvectors. $a(\omega_j)$ is the location vector modeled as

$$a\theta(\omega) = \frac{1}{\sqrt{L}} [1, e^{-j\omega Td}, \ldots, e^{-j\omega(L-1)Td}]^T \otimes \alpha\theta(\omega) \quad (4)$$

where

$$\alpha\theta(\omega) = \frac{1}{\sqrt{M}} [e^{-j\omega\tau 1,\theta}, \ldots, e^{-j\omega\tau M,\theta}]^T \quad (5)$$

and $\tau_{i,\theta} = (i-1)\tau\theta$, $\tau\theta = (\Delta/c)\sin\theta$

θ is the azimuth angle measured relative to array broad side

Δ is the sensor spacing

c is propagation velocity

$\tau\theta$ is the propagation delay from the array origin to the ith sensor for the source location θ.

then $a\theta(\omega)$ becomes

$$a\theta(\omega) = \frac{1}{\sqrt{L}} [1, e^{-j\omega Td}, \ldots, e^{-j\omega(L-1)Td}]^T \otimes \frac{1}{\sqrt{M}} [e^{-j\omega\tau 1,\theta}, \ldots, e^{-j\omega\tau M,\theta}]^T \quad (6)$$

The sensor spacing is 1/2 the propagation wavelength ($\frac{\Delta}{c} = \frac{1}{2}$) and the sampling interval is one (Td = 1). The above approach is known as a basis vector-based estimator where broad-band source representation subspace basis vectors are projected and combined.

The unit consists of eight specialized processors connected in an SIMD structure with linear and bi-directional links between one processor and its neighboring processors. Each of the processors has a bi-directional data bus with a RAM unit that stores the variables needed in the computations of the power method. Fig. 4. shows the power method architecture.

F. Generalized Processing Element (GPE) Suitable for DOA Problems

In previous sections, several SIMD structures were proposed with simple inter processor communications. Every processor should have the capability to communicate with its neighboring processors; the one on the right and the one on the left. A generalized processing element (GPE) has been designed and its block diagram is shown in Fig. 5.

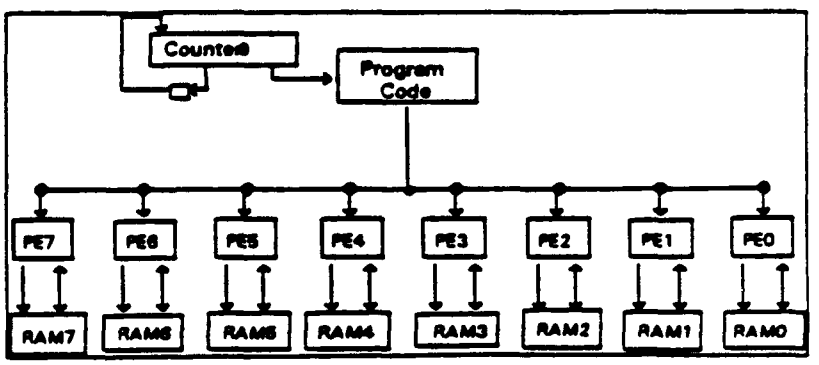


Fig. 4. The power method architecture.

The GPE has the following features,

1. Multiple Buses
2. AU capable of: addition, subtraction, multiplication, division, magnitude comparisons and performs complex functions: Sin, Cos and Log_n.
3. Eight general purpose File Registers
4. East/West Interconnections.
5. Micro Code Controls.
6. Address Generation and Selection.
7. Conditional/Unconditional Jumps.
8. Loops.

III. CONCLUSION

BASS-ALE algorithm has been simplified and converted into parallel/pipelined algorithm. An efficient architecture has been proposed for the covariance matrix that uses only eight PEs with minimal hardware requirements. The Householder unit uses mathematical shortcuts to enhance the algorithm and speedup the operations. It too uses an architecture of eight processors. And to add uniformity throughout the system, the QR architecture uses also eight processors. With minimal computation cycles required for the determination of the signal subspace dimension, a one processor setup is recommended. Similarly, an eight processor architecture is designed for the power method. The GPE offers the potential to perform multiple instructions simultaneously. Multiple buses, advanced AU operations and concurrent instructions all add to the composition of a faster processing power.

REFERENCES

[1] Arnab K. Shaw and Ramdas Kumaresan, "Estimation of Angles of Arrivals of Broadband Signals," ICASSP-87, pp.2296-2299,1987 IEEE.

[2] Guaning Su and Martin Morf, "Modal Decomposition Signal Subspace Algorithms," IEEE Trans. on Acoustics, Speech, and Signal Processing, VOL. ASSP-34, No. 3, pp. 585-602, June 1986.

[3] Björn Ottersten and Thomas Kailath, "Direction-of-Arrival Estimation for Wide-Band Signals Using the ESPRIT Algorithm," IEEE Trans. on Acoustics, Speech, and Signal Processing, VOL. 38, No. 2, pp. 317-327, February 1990.

[4] Kevin M. Buckley and LLoyd J. Griffiths, "Broad-Band Signal-Subspace Spatial-Spectrum (BASS-ALE) Estimation," IEEE Trans. on Acoustics, Speech, and Signal Processing, VOL. 36, No. 7, July 1988.

[5] M. Wax and T. Kailath, "Optimum Localizations ...," IEEE Trans. ASSP, VOL. ASSP-31, No. 5, pp. 1210-1218, Oct. 1983.

[6] A. Nehorai, G. Su, M. Morf, "Estimation of Time Difference of Arrivals For Multiple ARMA Sources By A Pole Decomposition Method," IEEE Trans. ASSP, VOL. ASSP-31, pp. 1478-1491, Dec. 1983.

[7] H. Wang and M. Kaveh, "Coherent Signal Subspace ...," IEEE Trans. ASSP, VOL. ASSP-33, No. 4, pp. 823-831, Aug. 1985.

[8] Guaning Su and Martin Morf, "The Signal Subspace Approach for Multiple Wide-Band Emitter Location," IEEE Trans. on Acoustics, Speech, and Signal Processing, VOL. ASSP-31, No. 6, pp. 1502-1521, December 1983.

[9] W. Phillips and W. Robertson, "A Systolic Architecture For The Symmetric Tridiagonal Eigenvalue Problem," International Conference on Systolic Arrays, pp. 145-150, 1988.

[10] Hao-Yung Lo and Jau-Ling Chen, "A Hardwired Generalized Algorithm for Generating the Logarithm Base-k by Iteration," IEEE Trans. on Computers, VOL. C-36, No. 11, pp. 1363-1367, November 1987.

[11] M. M. Jamali and S.C. Kwatra, "Development of Parallel Architectures For Sensor Array Processing Algorithms," Dept. of Electrical Engineering, The University of Toledo, Report No. DSPH-2, July 1992.

[12] Raja F. Tabar, "A Parallel-Pipelined Architecture For Broad-Band Direction-Of-Arrival Identification Problem," M.S. Thesis, Dept. of Electrical Engineering, The University of Toledo, in progress.

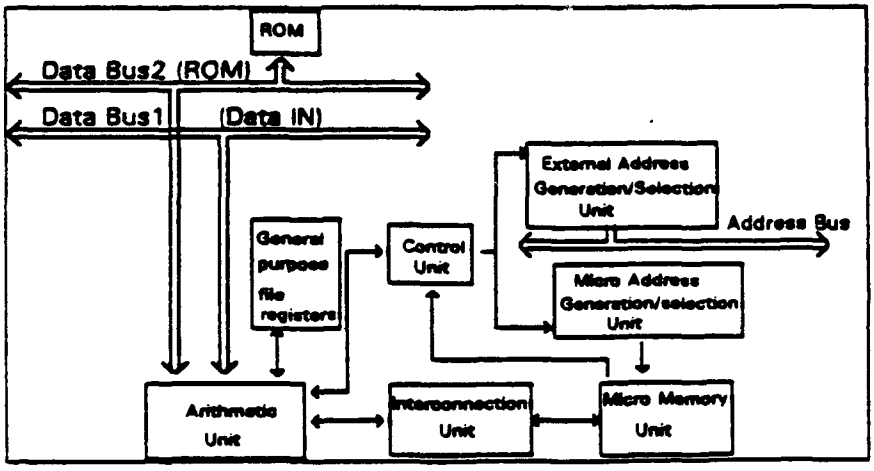


Fig. 5. Block diagram of the GPE.

AN ASIC Design of Genralized Covariance Matrix Processor for DOA Algorithms

M. M. Jamali, S. Ravindranath, S. C. Kwatra, A. G. Eldin
Department of Electrical Engineering
The University of Toledo
Toledo, OH 43606
419-537-2580

To be submitted to 1994 Internatational Symposium on Ciruits and Systems

AN ASIC Design of Genralized Covariance Matrix Processor for DOA Algorithms

M. M. Jamali, S. Ravindranath, S. C. Kwatra, A. G. Eldin
Department of Electrical Engineering
The University of Toledo
Toledo, OH 43606
419-537-2580

SUMMARY[1]

The estimation of the direction of arrival (DOA) in sensor systems has been one of the frequently considered problems in digital signal processing. The algorithms used to compute the DOA are based on the processing of the received signal and extracting the desired parameters to estimate the direction of arrival. The Multiple Signal Classification (MUSIC) [1] is a high resolution algorithm and has been widely used for computation of the DOA parameters. The approaches to this problem have been separated into the narrowband case which assumes that the signals can be considered to have only one frequency component and the broadband or wideband case in which the signal is considered to consist of a band of frequency components.

The goal of this work was to design an architecture which will be suitable for the narrowband and the wideband cases. It has been discovered that the narrowband MUSIC algorithm and the wideband algorithms BASS-ALE [2] and bilinear transformation [3] require similar computational modules such as the computation of the covariance matrix, eigenvalues and eigenvector computation using the Householder transformation and QR method and power method [4-5]. Since the required modules are identical, they can be generalized into one algorithm and generalized hardware can be developed. The generalized hardware will be suitable for both applications.

In the DOA applications, the data has to be collected by the sensors to compute the covariance matrix. In this study eight sensors and eight delay elements have been assumed and the hardware is designed accordingly. Eight sensors in the narrowband case will result in a 8*8 covariance matrix. Therefore all computations for DOA estimation will require manipulation of 8*8 matrices. It would be easier to map the algorithm on an archtitecture which has eight processing elements (PEs).

The first step in the estimation of DOA is the computation of the covariance matrix from the incoming signals. From the VLSI implementation point of view it is very appealing to design a combined covariance matrix processor which will be programmable and can be used for both narrowband and broadband algorithms. Such a combined processor has the advantage of being very cost effective and opens avenues to design a configurable system.

[1] This research is partly supported by ONR grant number N00014-91-J-1011

In this work three such algorithms which are very appropriate for the development of a dedicated system are considered and a combined covariance matrix processor is developed for them. The algorithms are the narrowband MUSIC algorithm [1] and the broadband BASS-ALE [2] and bilinear transformation algorithm[3]. The processor is designed to be compatible with an eight sensors and eight processor system which are placed on a processing board with each processor having its dedicated memory as shown in Figure 1. In this work the design and implementation of an ASIC chip for the processor is carried out. The processing board can be completed by using commercially available components for the memories.

PROCESSOR ARCHITECTURE

A block diagram of the combined covariance matrix processor [6] is shown in Figure 2. The architecture basically consists of the input loading stage, the arithmetic unit and the control units.The input loading stage consists of eight input latches for the **Y** vector and one for the **X** scalar and a load control unit which latches on the data at the appropriate clock pulse. The unit has two three bit counters and two 3 to 8 decoders. The latch counter is used to count the clock pulses and the decoder selects the appropriate latch according to the clock. One counter is used to latch the input data and is driven by the load clock. As shown in the flowcharts the load clock is received when the loading operation takes place. Once the data for one particular cycle is latched in, the load clock is disabled and the input clock which synchronizes the arithmetic operations is enabled. The input clock is used to drive the enable counter and the enable decoder which enables the appropriate buffer. The data from the latch is then placed on the internal data bus which is connected to the multipliers. As all the latches are connected to the same internal bus, a tristate buffer is used after the latch to prevent data corruption. The arithmetic unit has four multipliers, an adder, a subtractor and two accumulators. The control unit has three separate control modules for the three algorithms.

VLSI IMPLEMENTATION

The VLSI implementation of the processor described above involves a detailed design of the individual modules and transistor level optimization to provide a chip which can perform the required operations in the specified time frame. During the VLSI design various considerations such as the selection of multiplier and adder architectures and number representation were taken into account, and the chip design was carried out accordingly.

The VLSI simulation and implementation was carried out using two different software tools. First the behavioral simulation of the architecture was done on Powerview, the CAD package from Viewlogic Inc [7]. To perform a behavioral simulation of the proposed architecture, VHDL code was written for all the basic modules The processor was then constructed and it was sucessfully simulated. Secondly Mentor Graphics Generator Development Tools 5.3 [8] were used to perform the transistor and logic level simulation on the chip and conduct a timing analysis. The layout of the chip was generated and verified using the AutoCells feature in GDT.

A Led schematic of the combined covariance processor is shown in Figure 3. The results of the simulations are shown in the Figure 4. The inputs to the multipliers are mltina,

mltinb, mltinc and mltind. Two of these are the imaginary and real parts of the **X** input while the other two are the elements of the **Y** vector. The outputs of the complex multiplication are add and sub, while acc1 and acc2 give the values after accumulation. The input to the two accumulators from the memory are given by mem1 and mem2. The processor was simulated over two cycles and the multiplication and accumulation operations were verified.

The netlist was generated and an Lsim simulation was run on the netlist. The netlist for the processor was generated from the Led schematic. Then AutoCells was used to generate the layout of the processor. The layout was verified by simulating the netlist for the whole processor. The terminals were placed so that the routing to the pins can be done very easily. The data input terminals and the input control signals are placed at the top. The data bits (memory and processor out) are placed at the sides and the address bits are placed at the bottom. The total chip area is approximately 2200 x 5800 μm^2. The chip will fit in a 120 pin frame available through MOSIS. The layout of the processor is shown in Figure 5. The data pins are connected to the memory as shown in Figure 6. The address bits are supplied from the processor and a global reset pin is supplied so that the memory chip can be reset after one cycle of computations.

REFERENCES

[1] R.O. Schmith, "Multiple emitter location and signal parameter estimation," IEEE Trans. on Antennas and Propagation, Vol AP-34, No.3, PP. 276-280, Mar. 1986.

[2] Kevin M. Buckley and Lloyd J. Griffiths, "*Broad-band signal- subspace spatial-spectrum(BASE-ALE) estimation*", IEEE Trans. on Acoustics, Speech, and Signal Processing, VOL.36, No. 7, July 1988.

[3] Arnab K. Shaw and Ramdas Kumaresan, "*Estimation of angles of arrivals of broadband signals*", Proceedings of the Interantional Conference on Acoustic, Speech and Signal Processing , pp.2296-2299, 1987.

[4] M. M. Jamali, S.C. Kwatra, "Development of parallel architectures for sensor array processing algorithms. "Report No. DSPH-1, University of Toledo, 1991.

[5] M. M. Jamali, S.C. Kwatra, "Development of parallel architectures for sensor array processing algorithms. "Report No. DSPH-2, University of Toledo, 1992.

[6] R. Surya,"A Parallel and Pipelined Architecture for Estimation of Direction of Arrival using a Bilinear Transformation Method", M. S. Thesis, Department of Electrical Engineering, The University of Toledo, in progress.

[7] Powerview Viewlogic Reference Manuals, Viewlogic Inc. 1991.

[8] GDT, Mentor Graphics Corp. Software Manuals Version 5.3.1, 1992.

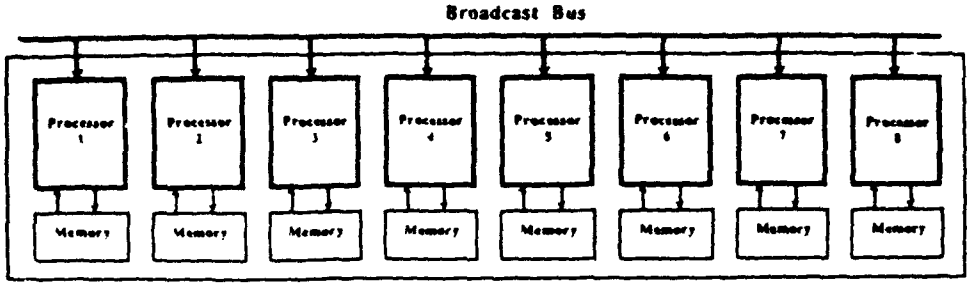


Figure 1 Processing board for the computation of covariance matrix.

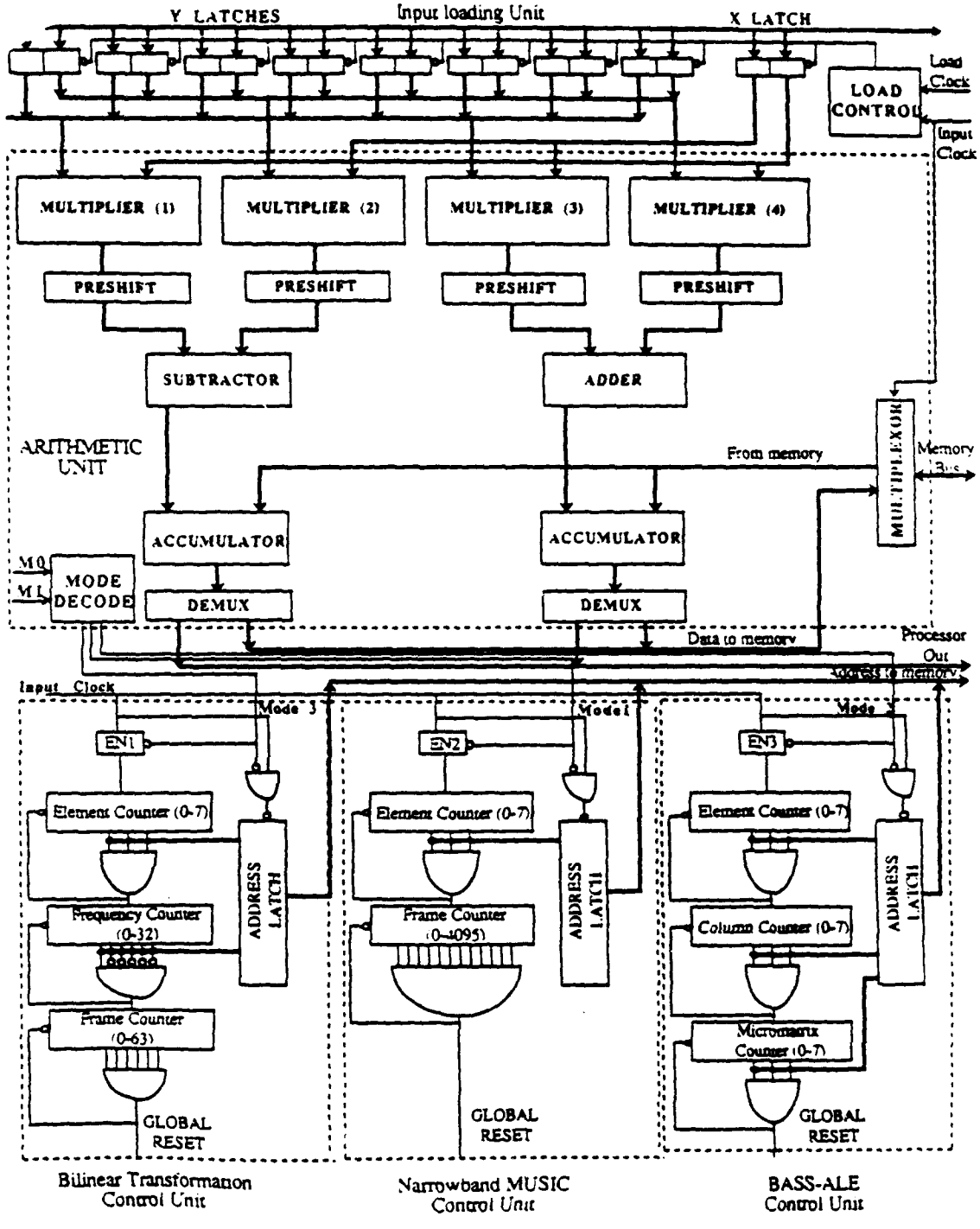


Figure 2: Generalized covariance matrix processing algorithm.

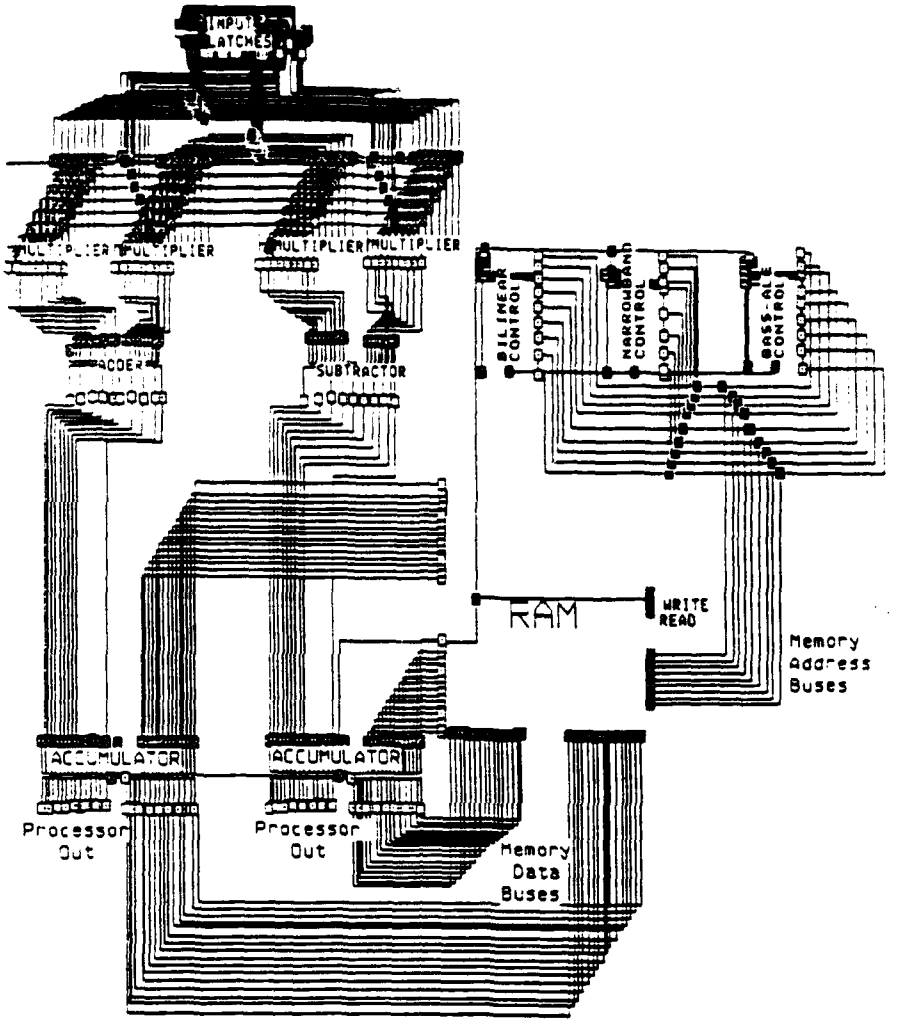


Figure 3: Led schematic of combined covariance matrix processor.

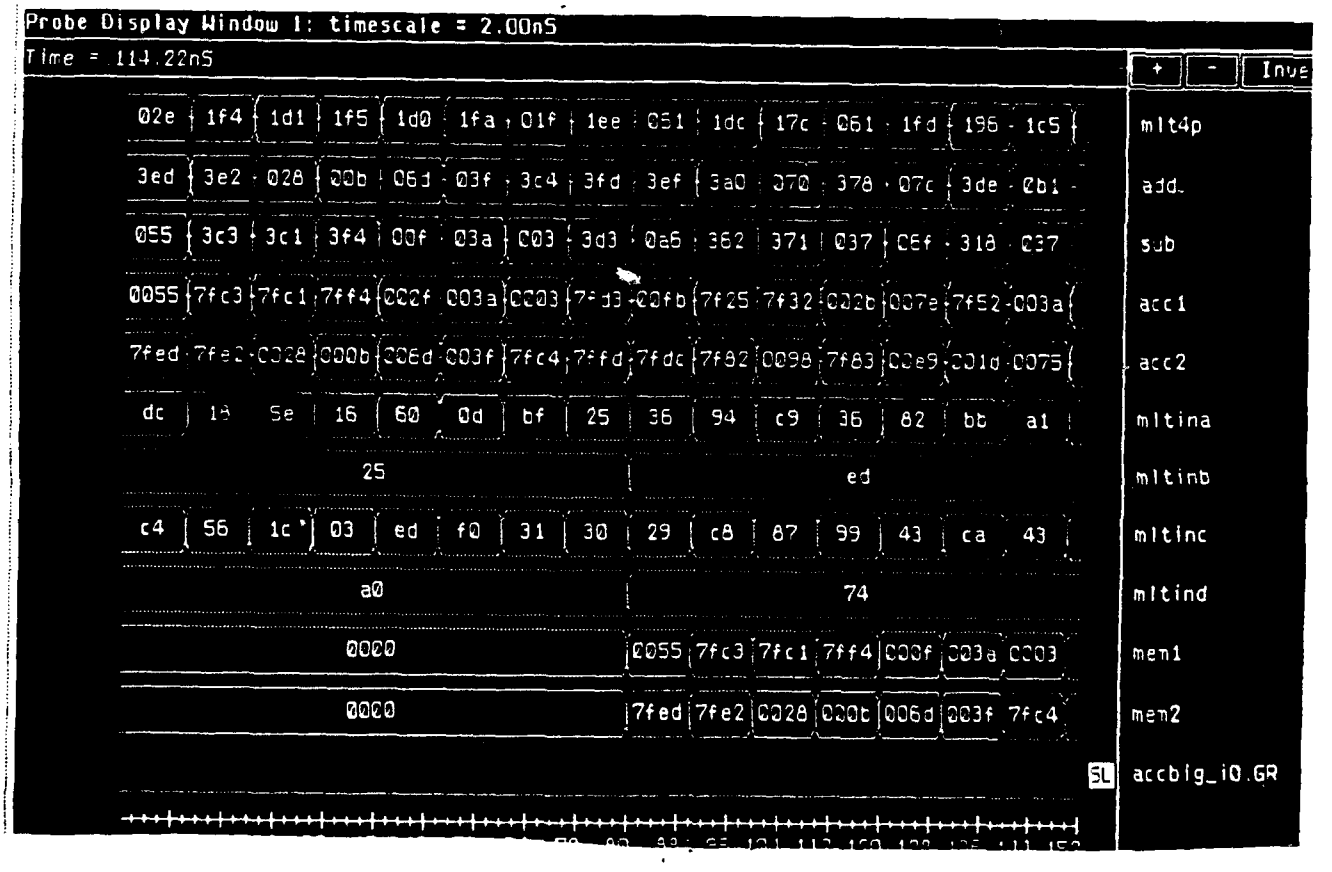


Figure 4: Lsim simulation results of the covariance matrix processor.

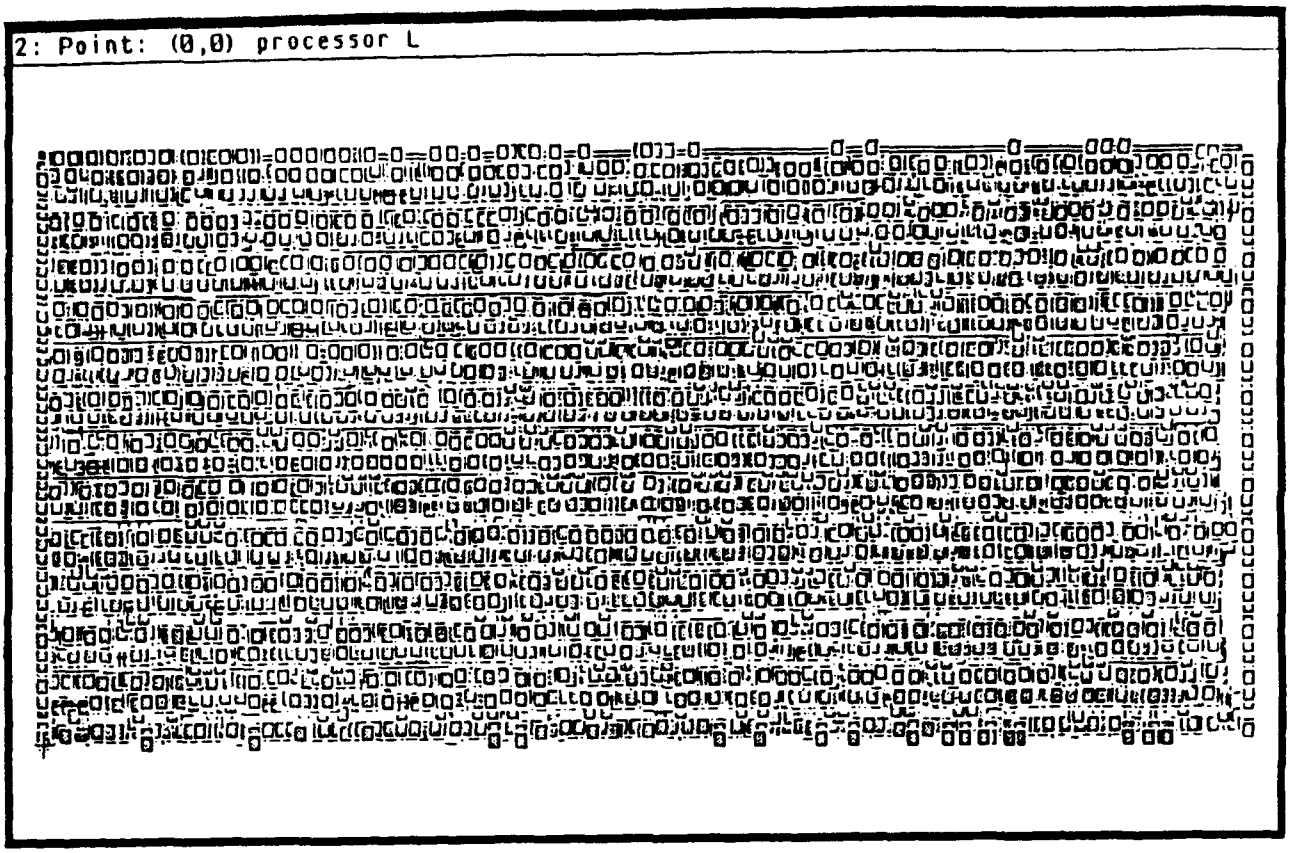


Figure 5: Layout generated using AutoCells for the covariance matrix processor.

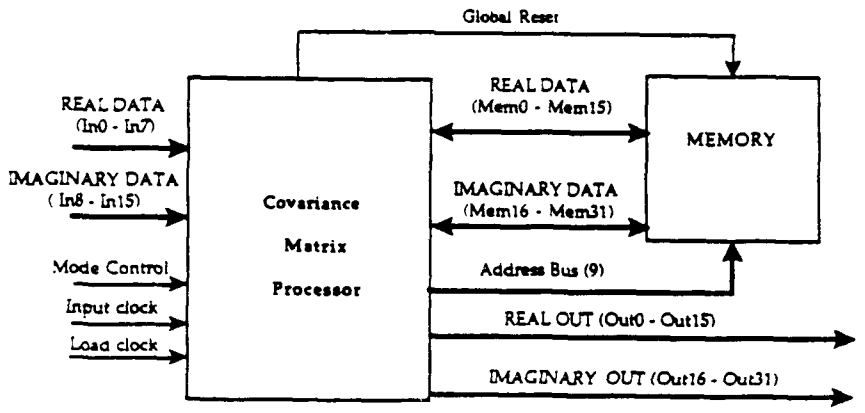


Figure 6: I/O Diagram of the covariance matrix processor

A Covariance Matrix Processor for DOA Algorithms

M. M. Jamali, S. Ravindranath, S. C. Kwatra
Department of Electrical Engineering
The University of Toledo
Toledo, OH 43606
419-537-2580

To be submitted to IEEE Journal of Oceanic Engineering

A Covariance Matrix Processor for DOA Algorithms

M. M. Jamali, S. Ravindranath, S. C. Kwatra
Department of Electrical Engineering
The University of Toledo
Toledo, OH 43606
419-537-2580

Abstract:

A generalized covariance matrix computation processor has been designed. The processor is suitable for DOA estimation using MUSIC algorithm for narrowband signal and BASS-ALE & bilinear transformation algorithms for wideband signals. The processor has been implemented using 0.8 micron CMOS technology and is suitable for real time processing.

Key Words: Sensor array processing, DOA estimation, Covariance matrix processor

I. INTRODUCTION

The estimation of the direction of arrival (DOA) in sensor systems has been one of the frequently considered problems in digital signal processing. The algorithms used to compute the DOA are based on the processing of the received signal and extracting the desired parameters to estimate the direction of arrival. The Multiple Signal Classification (MUSIC) [1] is a high resolution algorithm and has been widely used for computation of DOA estimation. The approaches to this problem have been separated into the narrowband case which assumes that the signals can be considered to have only one frequency component and the broadband or wideband case in which the signal is considered to consist of a band of frequency components [2-12].

The goal of this work was to design an architecture which is suitable both for narrowband and wideband cases. It has been determined that the narrowband MUSIC algorithm and the wideband algorithms BASS-ALE and bilinear transformation require similar computational

This research is partly supported by ONR grant number N00014-91-J-1011

modules such as the computation of the covariance matrix, eigenvalues and eigenvector computation using the Householder transformation and QR method and power method [13-15]. Since the required modules are identical, they can be generalized into one algorithm and generalized hardware can be developed. The generalized hardware will be suitable for both applications.

In the DOA applications, the data has to be collected by the sensors to compute the covariance matrix. In this study eight sensors and eight delay elements have been assumed and hardware is designed accordingly. Eight sensors in the narrowband case will result in a 8*8 covariance matrix. Therefore all computations for DOA estimation will require manipulation of 8*8 matrices. If eight Processing Elements (PEs) are used in the architecture it would be easier to map the algorithm on these eight PEs.

The first step in the estimation of DOA is the computation of the covariance matrix from the incoming signals. This is a preprocessing step which generates a correlation function from the data that is collected at the sensors. From the VLSI implementation point of view it is very appealing to design a combined covariance matrix processor which will be programmable and can be used for both narrowband and broadband algorithms. Such a combined processor has the advantage of being very cost effective and opens avenues to design a configurable system.

The design of a combined covariance processor is possible because the basic computations required in this stage are complex multiplications, accumulations and averaging, which are common to all three methods considered in this work. The main difference is in the control of the algorithms as they have different matrix sizes, number of matrices and the number of data samples. The processor is designed to be compatible with an eight sensors and eight processor system which are placed on a processing board with each processor having its

dedicated memory as shown in Figure 1. In this work [14-15] the design and implementation of an ASIC chip [16] for the processor is carried out. The processing board can be completed by using eight ASICs and commercially available memory ICs. The procedure involved in generating the covariance matrices is explained for the three algorithms in the following section.

II. THE COVARIANCE MATRICES

The sampled data obtained from the sensors is used to obtain the data covariance matrix. Since the covariance matrix is Hermetian, the computation of lower triangular matrix of covariance matrix is sufficient to get complete information of the full matrix. A generalized algorithm for the computation of covariance matrices for three algorithms have been designed and is suitable for mapping it on an eitght processor system. The C-like parallel algorithm for the computation of covariance matrix is given in Figure 2. The algorithm has three sections one for each of the three approaches and one common function which is called by every section. Three sections provide basic control sequence for the computation of the common function. Arithmetic computations are performed in the function and are common to all three approaches. This combined algortihm has been mapped on an architecture as shown in Figure 1. The architecture of the processing element is shown in Figure 3. Following sections describes execution sequence of three DOA approaches.

A. The Narrowband MUSIC algorithm

In the case of the narrowband MUSIC algorithm the covariance matrix generation is the the computation of 8 X 8 lower triangular matrix. For each computation of the DOA a total of 4096 such vector samples (frames) are collected. The covariance matrix is computed for each vector and the final matrix is obtained after taking the the average of 4096 computations.

First of all the whole system is reset using a global reset signal which clears all memory arrays and latches and initializes them to zero. The MUSIC algorithm is selected if the mode control is set to 00. The next step is to enable the frame counter (k) which counts the number of frames. A function is called to compute the matrix which is achieved by first loading the vector and performing computation in parallel.

To form one matrix a vector of 8 elements is needed from the sensors which are sampled simultaneously. Each processor will compute one column of the matrix by multiplying the 8 element signal vector by a scalar which is the element in the vector corresponding to that particular processor. For example the sixth processor in the array will multiply the vector by its sixth element. All eight processors will compute their respective columns in parallel. The computation of each matrix requires loading of the vectors and parallel computation. For each new frame the PE needs to load the incoming sampled data. The complete vector is broadcast to all the processors. The scalar element corresponding to each processor is individually routed to it. The control of the loading operation is handled by an external load counter (l_i) which synchronizes the loading of the data in all the processors. The loading operation is done over eight cycles during which one element is loaded for every cycle. In the first cycle the first element is loaded into the Y(1) latch of all processors and the X latch of the first processor. The latches in the processors are enabled by a decoder which is addressed by the 3 bit load counter. Once the loading is complete the arithmetic operations are started.

The next operation is of parallel computations which is controlled by the element counter. To complete the arithmetic operation and to generate the covariance matrix for complex numbers, it is necessary to perform four multiplications, an addition and a subtraction for each element. Apart from this there is an accumulation operation which is used to average all values over 4096 frames. Once the element counter is started, the appropriate data latch is enabled sending the output to the multiplier stage. The real and the imaginary parts of the

output are generated in parallel by performing four real number multiplications. This is done by the four eight bit multipliers in the arithmetic unit. A memory read operation is performed in parallel which will read the previously computed result and is added to the newly computed element. Once the accumulation is complete the address latch is enabled again and the result is written back to the memory. Then the frame counter is incremented and the operations are performed 4096 times. As explained above, overall 4096 such frames are accumulated. The sensor output is quantized into 8 bit real and imaginary parts and hence the word size becomes 16 bits after the multiplier stage. After the final accumulation the data becomes 28 (16+12) bits. This increases the chip area, data bus width and the memory requirements. To alleviate this problem the result is pre-shifted before accumulation by 6 bits.

B. Broadband BASS-ALE Algorithm

The BASS-ALE method is a broadband algorithm which uses the eigenstructure of a temporal covariance matrix and broadband source models to estimate the DOA. Similar to the MUSIC algorithm, the input vectors to the covariance stage are samples in the time domain. However for the BASS-ALE method operating with a system of eight sensors the input vector is a time delayed set of 64 samples. Eight samples are obtained from each sensor taken after a specific time delay. They are then stored in a delay array before the covariance processor stage, which gives a 64 element vector. The multiplication of a 64 element with its Hermetian yields a 64x64 matrix. A parallel and pipelined architecture for this procedure will consist of an array of 64 processors with each one computing one column of the resultant matrix. A new scheme has been proposed which allows the computation of the covariance matrix using an array of eight processors. An eight processor architecture is adopted as it is similar to the one proposed for the narrowband MUSIC and bilinear transformation algorithms.

The BASS-ALE algorithm can be selected by setting the mode signal to 01. The arithmetic operations are similar to the ones explained above. The major difference lies in the

controlling of the number of loops and the loading of the input latches. The control unit performs three nested loops for the BASS-ALE algorithm. The 64X64 matrix is split into 8 sub matrices each of which is 64 X 8 in dimension with the *i*th processor computing the *i*th submatrix. For example the 4th processor will compute the 4th submatrix which consists of the columns 25-32 in the covariance matrix. To simplify the control unit these submatrices are split up into eight 8 X 8 micromatrices. For each new column of the micromatrix the data has to be loaded into the PE. As before this is handled by an external load counter. The $(8k+i)$ th component of the vector is loaded to all the Y latches and the $(8i+j)$th scalar for that particular submatrix is loaded in the X latch. The arithmetic operations are the same as before with four multiplications, an addition and a subtraction. Simultaneously, the word is read in from the memory using *kji* of the counters as the address. The accumulating operation is then carried out and the result written back to the memory. Once the 8 elements of the column are calculated the processor computes the rest of the micromatrix and then each segment to finish one iteration of computations. The matrix is then accumulated over 512 loops and finally averaged, the global reset signal is enabled and the matrix is passed on for the computation of eigenvectors. The calculation of the covariance matrix for the BASS-ALE algorithms involves 64 times the number of computational operations when compared to the previous case. Hence to complete one full iteration the processor takes more time and to match the processor speed with the sensor speed a delay buffer before the processor stage is suggested.

C. The Broadband Bilinear Transformation Algorithm

This algorithm estimates the DOA of broadband sensor signals by using a simple bilinear transformation matrix. In the algorithm an approximation resulting from a dense and equally spaced array structure is used to combine the individual narrowband frequency matrices for coherent processing. This algorithm is non-iterative and does not require any initial estimates of the angles of arrival. Unlike the previous algorithms the input to the covariance matrix stage in this case is a result of an FFT operation. The input vector consists

of 8 elements in each frequency bin. The system uses data which are spread over 33 spectral bins so the covariance matrix processor needs to compute 33 covariance matrices. One covariance matrix is generated at each frequency bin and then averaged over 64 frames. The arithmetic operations are simlilar to previously described operations, but in this case as the averaging is done over only 64 frames the initial preshift by 6 bits is enough, and the shifting out after accumulation is not required.

III. PROCESSOR ARCHITECTURE

A block diagram of the combined covariance matrix processor [16] is shown in Figure 3. The architecture basically consists of the input loading stage, the arithmetic unit and the control units.The input loading stage consists of eight input latches for the **Y** vector and one for the **X** scalar and a load control unit which latches the data at the appropriate clock pulse. The load control unit has two three bit counters (latch and enable) and two 3 to 8 decoders. The latch counter is used to count the clock pulses and the decoder selects the appropriate latch according to the clock. The latch counter is used to latch the input data and is driven by the load clock. Once the data for one particular cycle is latched in, the load clock is disabled and the input clock which synchronizes the arithmetic operations is enabled. The input clock is used to drive the enable counter and the enable decoder which enables the appropriate buffer. The data from the latch is then placed on the internal data bus which is connected to the multipliers. As all the latches are connected to the same internal bus, a tristate buffer is used after the latch to prevent data corruption. The arithmetic unit has four multipliers, an adder, a subtractor and two accumulators. The control unit has three separate control modules for three algorithms.

IV. SIMULATION OF THE ARCHITECTURE

The behavioral simulation of the architecture was done on Powerview, the CAD package from Viewlogic Inc [17]. To perform a behavioral simulation of the proposed

architecture, VHDL code was written for all the basic modules The processor was then constructed from them. The Viewdraw schematic of the complete processor is shown in Figure 4. The data from the input latches is fed into the arithmetic unit which computes the complex number multiplication and gives the result to the accumulator. The other input of the accumulator is from the RAM. The memory result of the previous accumulation is read in using the address supplied from the address bus. Once the complete cycle of operations are performed the control unit generates the global reset signal which is used to place the output of the accumulator on the processor out pins. The Viewsim results of the processor simulation are shown in Figure 5.

V. VLSI IMPLEMENTATION

The VLSI implementation of the processor described above involves a detailed design of the individual modules and transistor level optimization to provide a chip which can perform the required operations in the specified time frame. During the VLSI design, various considerations such as the selection of multiplier and adder architectures and number representation were taken into account, and the chip design was carried out accordingly.

The VLSI simulation and implementation was carried out using Mentor Graphics Generator Development Tools 5.3 [18]. The GDT tools were used to perform the transistor and logic level simulation on the chip and conduct a timing analysis. The layout of the chip was generated and verified using the AutoCells feature in GDT. Following sections describes various stages of the processor.

A. The input loading stage.

The input stage consists of 9 sixteen bit latches and a load control unit. Eight of the input latches are used to hold the **Y** vector and the ninth one is loaded with the **X** scalar. The sixteen bit latch contains 8 bits for the real part and 8 bits for the imaginary part. The load

control provides 2 control signals. One is the latch control signal which dictates which latch is to be loaded at the particular time from the external broadcast bus. The other is the enable control which provides the signal to place the latch contents onto the processor data bus.

B. The arithmetic unit.

The arithmetic unit has the basic function of performing a complex multiplication and accumulation. It consists of four multiplier units, an adder, a subtractor and two accumulators. The multiplier [19] designed for the chip is a signed binary multiplier with a 7 X 7 array of full adders to compute the partial products. The final stage is a ripple adder which sums up the partial products. The sign bit is computed by a XOR gate which is fed by the sign bits of the two operands.

After the multiplying stage, there are four such products which are the result of the first stage of a complex number multiplication operation. These are the inputs to the adder and the subtractor. Ordinarily the subtraction of two of these operands will give the real part of the result, but in the generation of a covariance matrix, a vector is multiplied by its Hermetian, which is basically the transpose of its complex conjugates. Hence the operations are reversed and an addition is performed to obtain the real part of the result. The imaginary part can similarly be obtained by subtracting the two appropriate operands. The next stage is a 9 bit adder/subtractor followed by the accumulator. The accumulator consists of a 16 bit adder and a demultiplexing unit. One operand of the accumulator is the output from the previous adder/subtractor stage. The other is the previously stored result in the memory to which the newly computed value needs to be added. The output is connected to a demultiplexing stage which places the data either on the memory bus, writing the result back to the memory or on the processor out bus which signifies the completion of the processing of one frame of data. The demultiplexor is controlled by the global reset signal.

A major block which has been included in the schematic is the random access memory which is used to store the intermediate results of the operations. The required memory has been placed outside the chip so that a commercially available component can be used in conjunction with the processor ASIC to generate a reliable system. The memory is interfaced to the processor by a multiplexor as shown in the schematic of the processor. The data in and data out buses are connected to the multiplexor which is connected to the memory bus. The multiplexor is controlled by the input clock. The input clock has a duty cycle of 50% and hence can be used as a read/write signal. When the clock is high the processor reads from the memory and when the clock goes low the processor writes the output back to the memory. The size of the memory required for the operation is primarily dictated by the operations in the BASS-ALE algorithm which stores upto 2^9 elements during the computation of one covariance matrix. These elements are 32 bits wide including the real and imaginary components and hence require a RAM 16K bits in size. The RAM has a READ/WRITE signal, an enable and a reset signal which initializes all arrays to zero. For simulation purposes M model code was written for the RAM and the Lsim simulations were carried out in the multi-level mode.

C. The control units

The function of the control units is to generate the correct address for the retrieval of data from the memory during the accumulation stage. The control unit should also generate the global reset pulse once the processor finishes its cycle of operations. As most of the required control operation is to count the number of loops that the system has executed, the control unit consists mainly of counters.

The control unit for the narrowband MUSIC algorithm consists of two counters one of which is a 3 bit counter which upcounts to 7. This three bit counter is used to generate the address bits for the storage of the 8 different elements that are computed. The outputs of the 3 bit counter are fed to a 3 input AND gate which generates the clock pulse for the 12 bit frame

counter. The frame counter counts the 4096 loops that need to be executed during the accumulation process.

The control unit for the BASS-ALE algorithm has four counters, three of which are used to generate the address bits. The first counts the number of elements in the column, the second counts the column number in the micromatrix while the third keeps track of the micromatrix number in the submatrix. The 9 bit counter controls the numbers of frames which the processor needs to accumulate which in this case is 512.

The control unit for the bilinear transformation algorithm has a 3 bit element counter to count the element number in the column. The next one a 6 bit up counter, is used to count the 33 frequencies. Once it reaches 32, the logic circuitry (which is a NOR gate with an inverted MSB) resets it to zero and clocks the 6 bit frame counter. The frame counter counts the 64 frames that need to be accumulated.

All the modules were individually simulated. The individual modules were called instances and were placed into the top level processor cell in Led. The netlist was generated and an Lsim simulation was run on the netlist. The netlist for the processor was generated from the Led schematic. Then AutoCells was used to generate the layout of the processor. The layout was verified by simulating the netlist for the whole processor. The terminal were placed so that the routing to the pins can be done very easily. The data input terminals and the input control signals are placed at the top. The data bits (memory and processor out) are placed at the sides and the address bits are placed at the bottom. The total chip area is approximately 2200 x 5800 μm^2. The chip will fit in a 120 pin frame available through MOSIS. The layout of the processor is shown in Figure 6.

VI. CONCLUSIONS

A combined covriance matrix processor has been developed for three DOA algorithms, namely the narrowband MUSIC algorithm, the broadband BASS-ALE method and the bilinear transformation method. The processor has been simulated at the VHDL level using Powerview and then at the transistor level using GDT Lsim. The construction of the processor was done using the Lxcells utillity in GDT. Finally the processor was laid out using GDT Autocells.

VII. REFERENCES

[1] R.O. Schmith, "Multiple emitter location and signal parameter estimation," IEEE Trans. on Antennas and Propagation, Vol AP-34, No.3, PP. 276-280, Mar. 1986.

[2] Kevin M. Buckley and Lloyd J. Griffiths, "*Broad-band signal- subspace spatial-spectrum(BASE-ALE) estimation*", IEEE Trans. on Acoustics, Speech, and Signal Processing, VOL.36, No. 7, July 1988.

[3] Arnab K. Shaw and Ramdas Kumaresan, "*Estimation of angles of arrivals of broadband signals*", Proceedings of the Interantional Conference on Acoustic, Speech and Signal Processing , pp.2296-2299, 1987.

[4] H. Wang and M. Kaveh, "Coherent Signal Subspace processing for the detection and estimation of angels of arrival of multiple wide-band sources," IEEE Trans. ASSP, VOL. ASSP-33, No. 4, pp. 823-831, Aug. 1985.

[5] Guaning Su and Martin Morf, "Modal Decomposition Signal Subspace Algorithms", IEEE Trans. on Acoustics, Speech, and Signal Processing, VOL. ASSP-34, No. 3, pp. 585-602, June 1986.

[6] Björn Ottersten and Thomas Kailath, "Direction-of-Arrival Estimation for Wide-Band Signals Using the ESPRIT Algorithm", IEEE Trans. on Acoustics, Speech, and Signal Processing, VOL. 38, No. 2, pp. 317-327, February 1990.

[7] M. Wax and T. Kailath, "Optimum Localizations of multiple sources by passive arrays," IEEE Trans. ASSP, VOL. ASSP-31, No. 5, pp. 1210-1218, Oct. 1983.

[8] B. Porat and B. Frienlander, "Estimation of Spatial and Spectral Parameters of Multiple Sources," IEEE Trans. on Information Theory, VOL. IT-29, pp. 412-425, May 1983.

[9] A. Nehorai, G. Su, M. Morf, "Estimation of Time Difference of Arrivals For Multiple ARMA Sources By A Pole Decomposition Method", IEEE Trans. ASSP, VOL. ASSP-31, pp. 1478-1491, Dec. 1983.

[10] M. Wax et al, "Spatio-Temporal Spectral analysis by eigenstructure methods," IEEE Trans. on ASSP. VOL. ASSP-32, No. 4, Aug. 1984.

[11] Guaning Su and Martin Morf, "The Signal Subspace Approach for Multiple Wide-Band Emitter Location", IEEE Trans. on Acoustics, Speech, and Signal Processing, VOL. ASSP-31, No. 6, pp. 1502-1521, December 1983.

[12] Alle-Jan van der Veen and Ed. F. Deprettere, "Parallel VLSI Matrix Pencil Algorithm for High Resolution Direction Finding", IEEE Trans. on Signal Processing, VOL. 39, No. 2, pp. 383-394, February 1991.

[13] W. Phillips and W. Robertson, "A Systolic Architecture For The Symmetric Tridiagonal Eigenvalue Problem", International Conference on Systolic Arrays, pp. 145-150, 1988.

[14] M. M. Jamali, S.C. Kwatra, "Development of parallel architectures for sensor array processing algorithms. "Report No. DSPH-1, University of Toledo, 1991.

[15] M. M. Jamali, S.C. Kwatra, "Development of parallel architectures for sensor array processing algorithms. "Report No. DSPH-2, University of Toledo, 1992.

[16] R. Surya,"A Parallel and Pipelined Architecture for Estimation of Direction of Arrival using a Bilinear Transformation Method", M. S. Thesis, Department of Electrical Engineering, The University of Toledo, in progress.

[17] Powerview Viewlogic Reference Manuals, Viewlogic Inc. 1991.

[18] GDT, Mentor Graphics Corp. Software Manuals Version 5.3.1, 1992.

[19] Ma, G.K, and Taylor F.J., "*Multiplier policies for Digital Signal Processing*", IEEE ASSP Magazine January 1990.

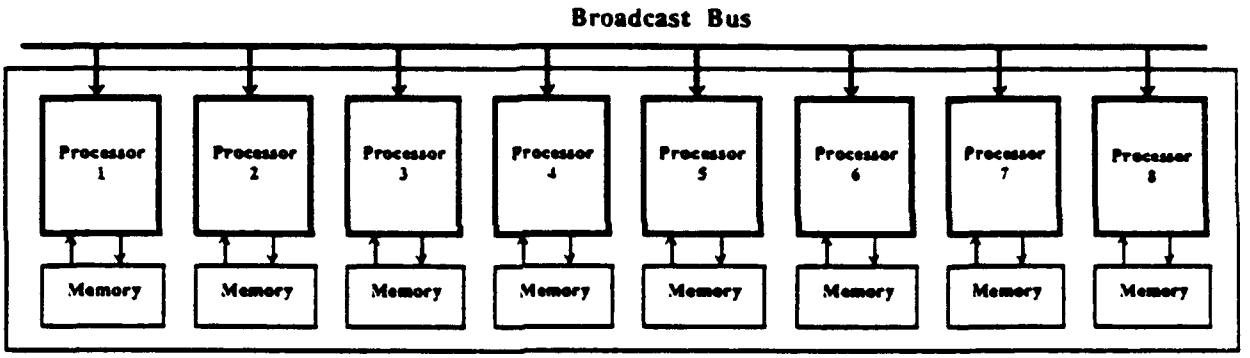


Figure 1 : Processing board for the computation of covariance matrix

```
/* generalized covarinace matrix processing algorithm */
main ( )
{
initialize all arrays and counters;
switch (val) {
case 1:
    enable narrowband control unit;
    while (frame_counter_k <=4095) {
    void compute_matrix-operations
    ++frame_counter_ k; }
break;
case 2:
    enable BASS-ALE control unit;
    while(frame_counter_n<=511)
    {
        while(micromatrix_counter_k<=7) {
                while (column_counter_j<=7) {
                void compute_matrix-operations
                ++column_counter_j; }
        ++micromatrix_counter_k; }
    ++frame_counter_n;
    }
break;
case 3:
enable bilinear transformation control unit;
    while(frame_counter_k<=63) {
        while(frequency_counter_j<=32) {
        void compute_matrix-operations;
        ++frequency_counter_j; }
    ++frame_counter_k; }
break;
}
}
void compute_matrix-operations;
{
        while ( load_counter_i <=7) {
        par load i th component in Y latch;
        par load scalar in X latch of the i th processor;
        ++load_counter_i; }
        while (element_counter_i <=7) {
        par complex multiplications;
        par shift right 6 bits;
        par accumulate;
        ++ element_counter_i; }
}
```

Figure 2: Generalized covarinace matrix processing algorithm

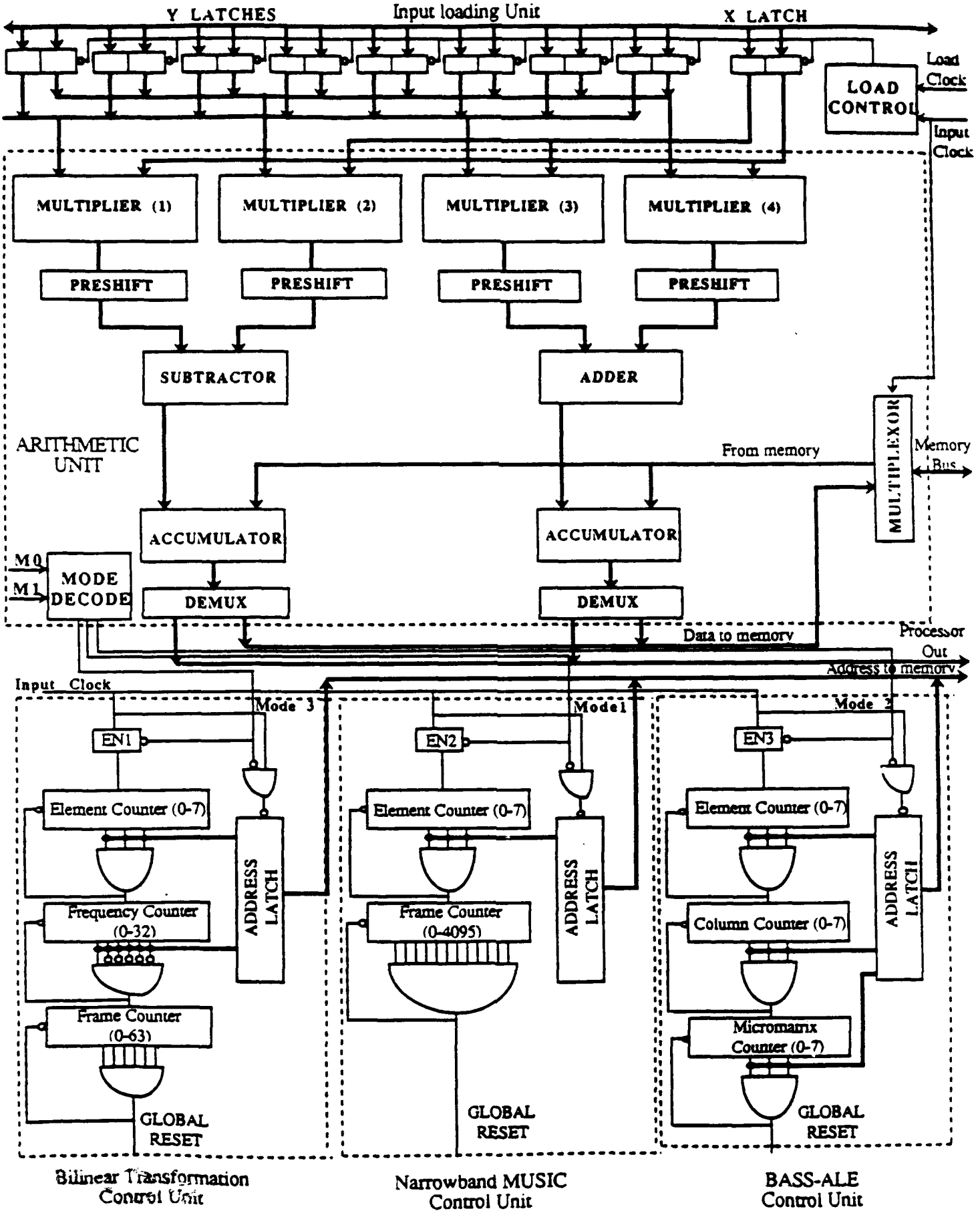


Figure 3: Block diagram of combined covariance matrix processor

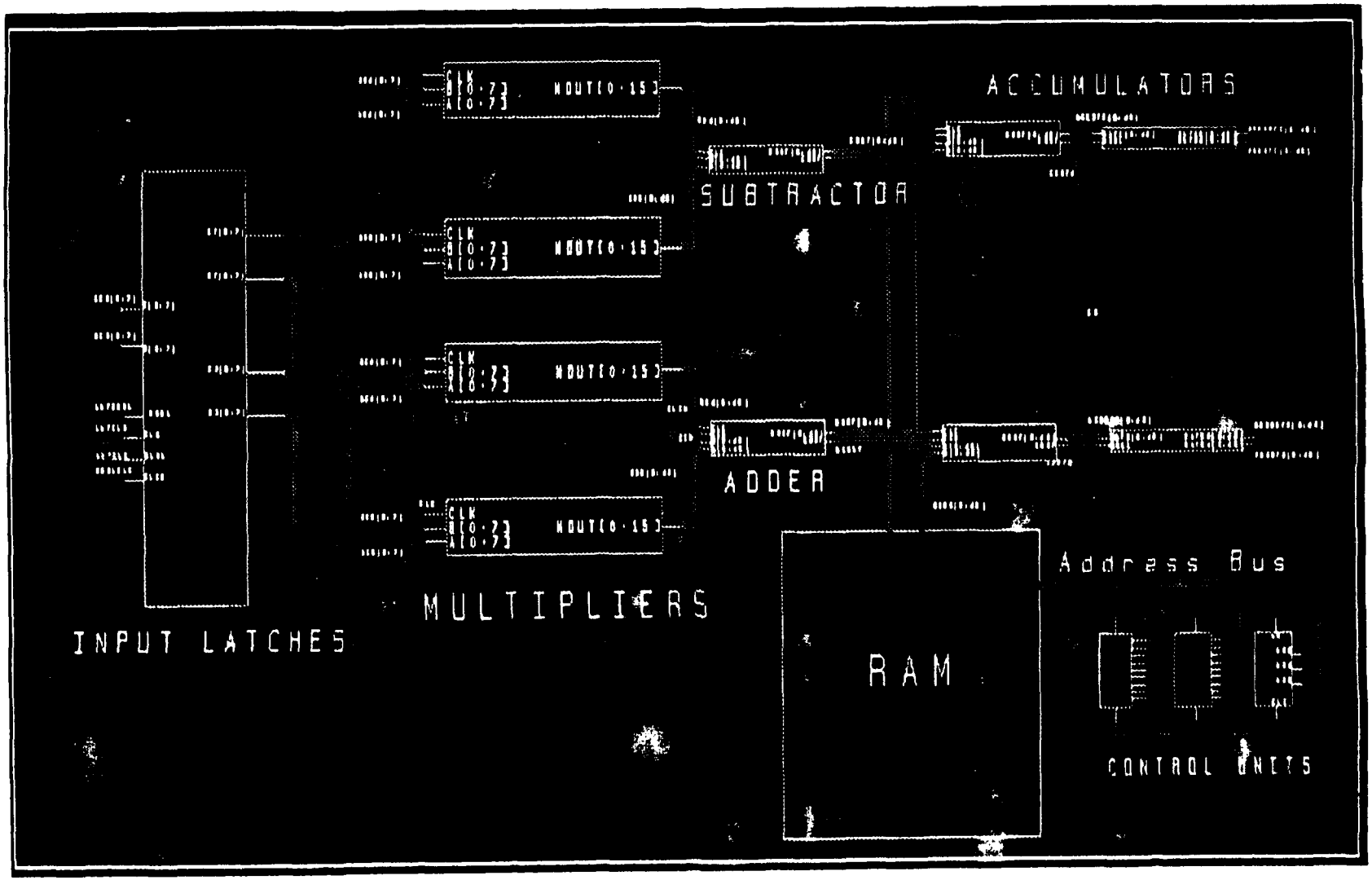


Figure 4 Viewdraw top level Schematic of Combined covariance matrix processor

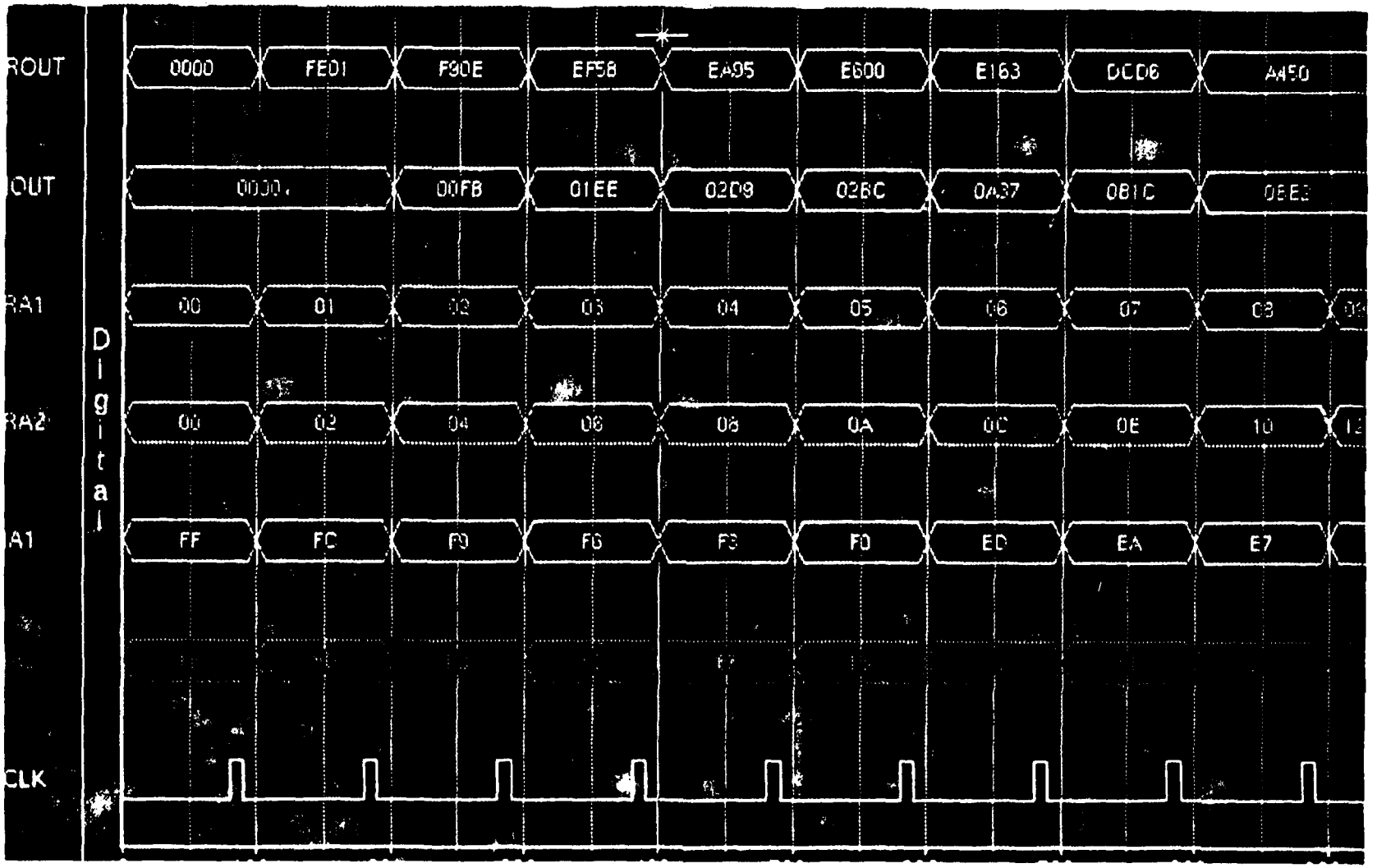


Figure 5 . Viewsim results for simulation of covariance matrix processor

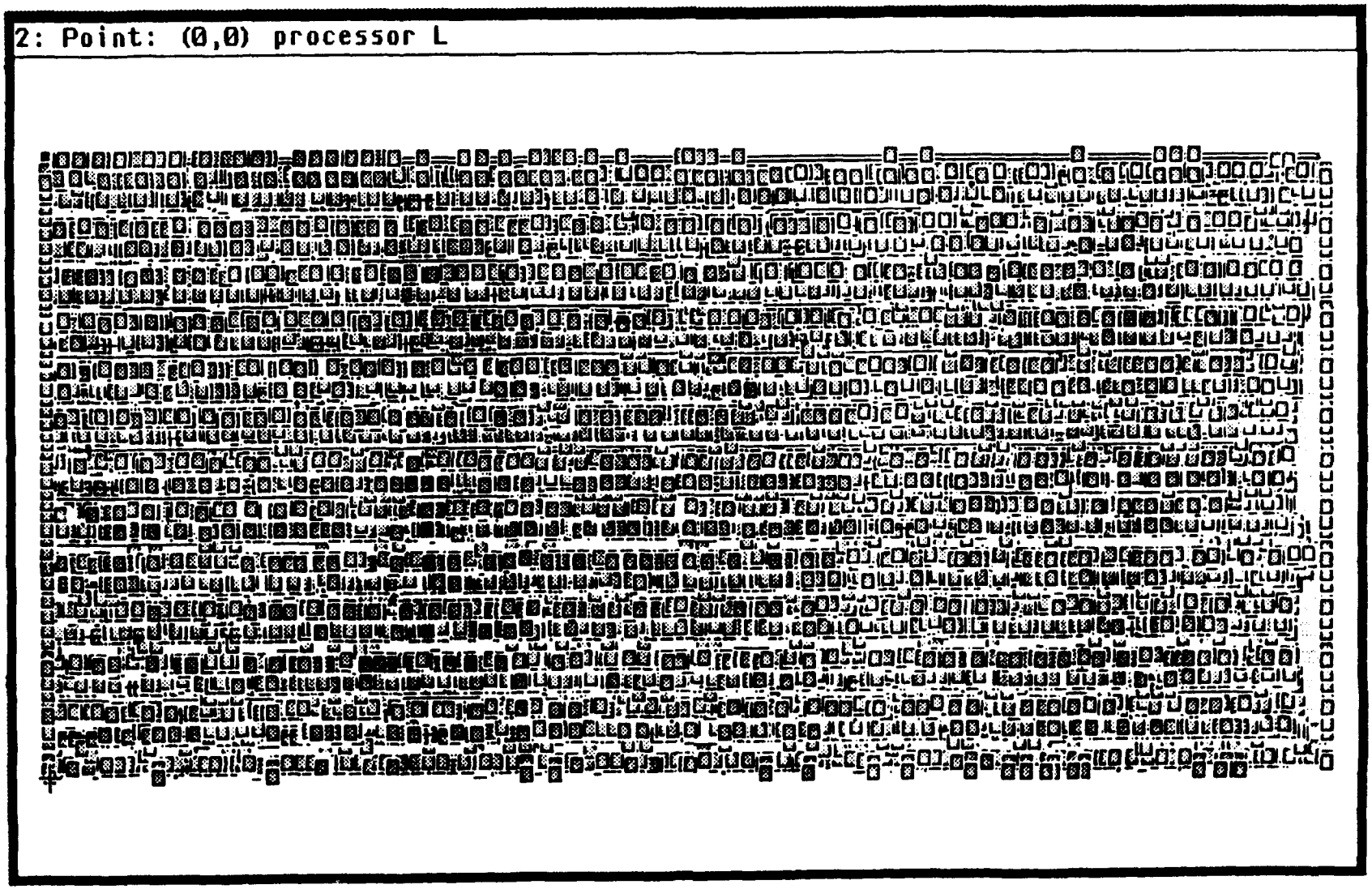


Figure 6 Layout generated using AutoCells for the covariance matrix processor